TAMPERE UNIVERSITY OF TECHNOLOGY

SYEDA SAKIRA HASSAN

BIOPROCESS OPTIMIZATION USING MACHINE LEARNING METHODS

Master of Science Thesis

Examiners: University Lecturer Heikki Huttunen
Dr.(Tech) Tommi Aho
Examiners and topic approved by
the Faculty council of Computing and Electrical Engineering on 08.05.2013

ABSTRACT

TAMPERE UNIVERSITY OF TECHNOLOGY
Master's Degree Programme in Information Technology
HASSAN, SYEDA SAKIRA: Bioprocess Optimization using Machine Learning Methods
Master of Science Thesis, 54 pages
October 2013
Major: Signal Processing
Examiners: University Lecturer Heikki Huttunen, Dr.(Tech) Tommi Aho
Keywords: optimization, data analysis, yield prediction, model assessment

In bioprocess development, the need for optimization is to achieve improvements in the productivity as well as in the quality of the product. This involves acquiring an overview of dataset associated with different process runs, identifying primary control parameters, and determining a useful control direction. Hence, the use of several data analysis approaches to explore optimization possibilities can be very valuable in bioprocess development.

In this thesis, multiple linear regression, Lasso regression, and artificial neural networks were used for modeling a bioprocess dataset. As a case study, we used the data obtained from a statistical culture media optimization experiment for microbial hydrogen production. Apart from the linear models, dataset were transformed to build the quadratic multiple linear regression and Lasso models. In addition, two-layer and three-layer artificial neural networks models were also developed. In order to predict the maximum achievable hydrogen production yield, a genetic algorithm was used to optimize the parameters of the developed models. The prediction accuracy and the maximum achievable hydrogen yield by Lasso and artificial neural networks models were benchmarked against those of the multiple linear regression. All the three methods were capable in providing a significant model for the culture media optimization. However, the performance of the quadratic multiple linear regression to fit the examined data was not adequate. In this case, the correlation between the observed and predicted yield was 0.37. The modeling was still successful with the quadratic Lasso model (0.82). The performances of two artificial neural network models outperformed the others. According to artificial neural networks, the correlations between the observed and predicted yield were 0.92 for two-layer and 0.91 for three-layer models. With the help of genetic algorithm, the maximum achievable hydrogen yield was 2.24 mol-H_2/mol-glycerol$_{consumed}$ for the linear multiple linear regression model. On the other hand, the results obtained from the Lasso and artificial neural networks models were closer to the highest experimental observation. Thus, we found that both lasso regression and artificial neural networks were pertinent to this kind of bioprocess data.

PREFACE

This Master's thesis work is conducted in the Department of Signal Processing of Faculty of Computing and Electrical Engineering at Tampere University of Technology (TUT).

I am utmost grateful to my thesis supervisors Dr.(Tech) Tommi Aho and University Lecturer Heikki Huttunen for introducing me with the research topic and providing comments, advices and suggestions during different phases of this research. I would like to thank them once more for the excellent guidance, encouragement and support throughout my study and research. My greatest gratitude goes to Professor Olli Yli-Harja for giving me the opportunity to work in the field of Computational Systems Biology (CSB).

I would also like to sincerely thank MSc. Rahul Mangayil for providing the invaluable data for this work. I am very grateful to MSc. Muhammad Farhan for his support and cooperation at work and sharing knowledge and experiences in the research project. My special thanks to my colleague and best friend MSc. Laura Järvenpää for her guidance, inspiration, and reviewing the thesis.

Finally, I would like to dedicate warm thanks to my family who have supported and encouraged me throughout my studies. Specially, my warmest thanks to my husband Sharif for all the support and patience during the joys and frustrations along the research work.

Syeda Sakira Hassan
Tampere, 21.09.2013

CONTENTS

1. Introduction 1
   1.1 Related works 1
   1.2 Objective of the thesis 2
   1.3 Structure of the thesis 3
2. Modeling methods 4
   2.1 Linear regression 4
   2.2 Artificial neural networks 8
3. Model assessment and optimization 14
   3.1 Model assessment 14
   3.2 Optimization 18
      3.2.1 Framework of genetic algorithm 19
      3.2.2 Major components in genetic algorithm framework 20
      3.2.3 Examples of genetic algorithm 24
4. Case study materials 28
5. Experiments and results 32
   5.1 Yield prediction 32
      5.1.1 Multiple linear regression 32
      5.1.2 Lasso 34
      5.1.3 Artificial neural networks 36
   5.2 Performance of the models 39
   5.3 Optimization of the yields 41
   5.4 Analysis of the results 43
6. Discussion and conclusion 48
References 50

LIST OF SYMBOLS AND ABBREVIATIONS

X	Independent variable
y	Dependent variable
$\hat{y}$	Predicted response
β	Regression coefficient
ϵ	error term
$(.)^T$	Transpose operator
$(.)^{-1}$	Inverse operator
$\|.\|$	Euclidean distance or $L^2 - norm$
$\|.\|_1$	$L^1 - norm$
$\|.\|_p$	$L^p - norm$
λ	Regularized parameter
t	Tuning parameter
I	Identity matrix
b	Bias
w_{ij}	Weight associated from node i to node j
a, d, c	Constants
η	Learning rate
δ_j	Local gradient
μ	Mean
Z	Linear combination of multiple weighted inputs with bias in a network
δ	Loss or cost function
σ	Standard deviation
ρ	Coefficient of correlation

p_c	Crossover rate
p_m	Mutation rate
$sin(.)$	Sinusoidal function
g/L	Gram per liter
mol-H_2/mol-glycerol$_{consumed}$	Mass of H_2 produced by consuming per mass of glycerol
ANN	Artificial neural network
BFGS	Broyden, Fletcher, Goldfarb, and Shanno algorithm
$C_4H_{11}NO_3$.HCl	Trypton
$C_{12}H_7NO_4$	Resazurin
$C_2H_3NaO_2.3H_2O$	Sodium acetate trihydrate
CV	Cross validation
DNA	Deoxyribonucleic acid
H_2	Hydrogen
K_2HPO_4	Dipotassium phosphate
KCl	Potassium chloride
GA	Genetic algorithm
KH_2PO_4	Monopotassium phosphate
LASSO	Least absolute shrinkage and selection operator
LOOCV	Leave-one-out cross validation
$MgCl_2.6H_2O$	Magnesium chloride hexahydrate
MLR	Multiple linear regression
MLP	Multilayer perceptron
$Na_2S_2O_4$	Sodium dithionite
NH_4Cl	Ammonium chloride
SUS	Stochastic universal sampling

1. INTRODUCTION

With the advent of technology, industrial biotechnology has been emerging in everyday life, from food to health care, from agriculture to products. With the aid of modern computers, a variety of process control and data analysis platforms and tools are available. In the field of biotechnology, a bioprocess control is defined as providing a near optimal environment for processes that use biological components or living organisms, such as yeast, enzymes and microorganisms to obtain the desired products. The desired products can be for instance, active pharmaceutical ingredients such as vaccine, health-care products such as vitamins, nutrients such as amino acid, fine chemicals and bulk chemicals such as alcohol. The aim of bioprocess optimization is to achieve improvements in the outcome of processes and in the quality of end products. These improvements require the right concentrations of nutrients to the medium as well as controlling important internal process parameters (such as pH, temperature). The scope of bioprocess development thus includes the need for data analysis.

In recent years, researchers have become increasingly interested in finding alternative renewable energy sources due to the limited resource of fossil fuels and global warming awareness. An excellent alternative to fossil fuel is biohydrogen (H_2), as it is considered to be non-polluted and non-exhaustible [1]. It can be obtained both from cultivation and waste organic materials [2, 3]. As an example, crude glycerol is a byproduct produced during biodiesel manufacturing process. It is used for hydrogen production using microbial processes [2, 4]. Researchers have investigated that crude glycerol can be utilized effectively for hydrogen production [5]. Thus, to increase the economic value of byproducts, the improvement in hydrogen production is becoming a promising application area in the biotechnological field.

1.1 Related works

The history of applying biotechnology started around 6000 B.C., when people developed the knowledge of making fermented foods and alcoholic beverages. However, the process was not explained properly until 1857, when Louis Pasteur ascertained that yeast is a living cell that ferments sugar to alcohol [6]. Methods such as factorial design, design of experiments, and response surface methodology were developed during early 1900s to investigate the mathematical relationships between input and

output variables of a process [7]. It was not until recent years that these methods were widely applied in the development of biotechnological processes. A simplified bioprocess is shown in Figure 1.1. By using the aforementioned methodologies, researchers investigate the input variables from which well-defined output responses are generated. The output responses can be for example, product yield or productivity. It is often difficult to discover the interactions between the input variables that influence the output responses, since a typical bioprocess development includes various sequential steps. For instance, in most bioprocesses, products are recovered at downstream stage where additional variables are supplied to purify the product. The input variables in the upstream stage also add further complexity to the bioprocess development. Therefore, combining all input variables in a bioprocess modeling may end up in a model with incomprehensible number of interacting or noninteracting terms. These terms may or may not have any effect on the specific outputs.

Figure 1.1. *A block diagram of a simple bioprocess.*

The design of experiments methodology has been used extensively by providing powerful and efficient ways to optimize bioprocesses. In a fermentative hydrogen production, for example, Pan et al. had studied the effect of 8 variables on hydrogen yield. As an initial step, the authors screened 3 key important variables using Plackett-Burman design [8]. They also used response surface methodology to depict the results in a contour plot where the optimum was clearly visualized. However, the study of Nagata and Chu showed that response surface methodology was not always guaranteed to identify the optima. They proposed another alternative solution for the conventional approach. In fact, they showed that the higher modeling capability of artificial neural networks and finding optimum solution by genetic algorithms were performed better than the standard response surface methodology [9].

1.2 Objective of the thesis

The conventional design of experiments is not fully explored. With full factorial design, for instance, all possible combination of variables effect on response can be investigated. With 2 variables, this method requires 2^2 runs of experiments. As the number of variables increases, the number of runs increases geometrically. Thus, this design may become impractical when the effects of a large number of variables

are to be studied. Furthermore, the response surface methodology may not always find the optima due to poor modeling capability of the quadratic model [10]. Hence, the possibility of using non-statistical approaches may provide alternative solutions to this traditional methodology.

This thesis explores the new optimization possibilities in bioprocess development by utilizing dataset obtained from the design of experiments. Moreover, the prediction capabilities of the models developed by several data analysis approaches are also analyzed.

1.3 Structure of the thesis

We have organized the rest of this thesis in the following way. **Chapter 2** provides a brief introduction to prediction methods. In **Chapter 3**, we describe the algorithms to assess the performance of the developed models. Apart from model assessment, this chapter also provides a brief introduction to optimization technique.

In **Chapter 4**, we present the materials which are considered for the experiment. In **Chapter 5**, models are developed for the given material. The performance of each model is assessed by using the algorithms explained in **Chapter 3**. In addition to evaluation of model performances, the predicted responses are optimized. Finally, **Chapter 6** concludes this work and proposes a future research direction.

2. MODELING METHODS

Prediction problems are often encountered in bioprocess modeling. They require the identification of important parameters as well as predicting the parameter values from a dataset. Such problems may occur in various disciplines for instance, food, biomedical, and biofuels industries [11].

Viewing from data analysis perspective, the main difficulty in such problems is to cope with the characteristics such as multicollinearity and ill-posed nature embedded in the original dataset. Therefore, feature selection and estimation of parameters are essential for modeling methods. Although several popular prediction methods exist, this thesis is limited to linear regression methods and artificial neural networks. In this chapter, we will briefly introduce a popular regularized least squares technique - *Lasso* and artificial neural networks.

2.1 Linear regression

Linear regression is an approach to model the relationships between the dependent variable, denoted as $\mathbf{y} \in \mathbf{R}^{n\times 1}$ and a combination of one or more explanatory or independent variables, denoted as $\mathbf{X} \in \mathbf{R}^{n\times p}$, where n is the number of observations and p is the number of variables. With one independent variable, it is known as the *simple linear regression*. If there are more than one independent variable, then the regression model is called the *multiple linear regression* [12]. Linear regression can also be represented by

$$y = f(X) + \epsilon \tag{2.1}$$

That is, $\mathbf{y}$ is a linear function of $\mathbf{X}$. Here, ϵ is an error term, which is an unobserved random variable that adds noise to the relationship between dependent variable and independent variables. The function f is called the *linear predictor function*, which is a linear combination of a set of coefficients and independent variables. The coefficients are known as the *regression coefficients*. Equation (2.1) can be expressed as

$$y = \beta_0 + \sum_{i=1}^{p} x_i\beta_i + \epsilon \tag{2.2}$$

where β_0 is the intercept, also known as *bias*. In Equation (2.2), $\beta = (\beta_0, \beta_1, \ldots, \beta_p)^T$ are coefficients which are unknown for the given p-dimensional inputs $X = (x_1, x_2, ..., x_p)^T$.

The linear model is obtained by estimating the unknown regression coefficients from a given dataset. The most popular method for this purpose is the *least squares fitting* [13]. Rewriting Equation (2.2) in vector format, we get

$$y = X\beta + \epsilon \tag{2.3}$$

In the least squares method, the coefficients vector β is chosen which minimizes the residual sum of squares. Thus, a unique solution is given by

$$\hat{\beta} = (X^T X)^{-1} X^T y \tag{2.4}$$

It can be shown that this solution minimizes the residual. That is

$$\hat{\beta} = \arg\min_{\beta} \|y - X\beta\| \tag{2.5}$$

where $\|.\|$ is the standard L^2-*norm* in the n-dimensional Euclidean space $\mathbf{R}^n$. For a real number $p \geq 1$, L^p-*norm* or p-*norm* of X can be defined as $\|X\|_p = (\|x_1\|^p + \|x_2\|^p + \ldots + \|x_n\|^p)^{1/p}$. When p is omitted, then the norm is L^2-*norm*.

Although least squares approach is easily interpretable and it can well approximate the linear behavior of the given dataset, the solutions are not always satisfactory for the following reasons:

1. The least squares method is sensitive to outliers.

2. The method may not provide a unique solution when the number of variables is larger than the number of data samples ($p \gg n$). In this case, the covariance matrix $X^T X$ in Equation (2.4) is singular and thus cannot be inverted.

3. The prediction accuracy may sometimes lead to poor performance because of interdependencies among explanatory variables.

4. If the relationships between the dependent and the explanatory variables are nonlinear, least squares method does a poor job in modeling.

The first three ill-posed problems listed above can be mitigated by a *regularization* technique [14–16]. Regularization is a technique which shrinks the coefficients by imposing a penalty on the size of the coefficients. Although many regularization algorithms have been proposed, *ridge regression* or *Tikhonov regularization* [17, 18] and *Lasso (Least Absolute Shrinkage and Selection Operator)* [16] are considerably well-known methods. The idea is to minimize the variance by compromising little bias. Both methods minimize the residual sum of squares and a penalized term. Therefore, Equation (2.5) becomes

$$\hat{\beta}_{ridge} = \arg\min_{\beta} \left\{ \|y - X\beta\| + \lambda \|\beta\| \right\} \tag{2.6}$$

for ridge regression and

$$\hat{\beta}_{lasso} = \arg\min_{\beta} \left\{ \|y - X\beta\| + \lambda \|\beta\|_1 \right\} \tag{2.7}$$

for Lasso in *Lagrangian form*. Here, $\lambda \geq 0$ is the *regularized parameter* which limits the size of regression coefficients. Another equivalent formulation of Equation (2.6) and Equation (2.7) is

$$\hat{\beta}_{ridge} = \arg\min_{\beta} \|y - X\beta\| \text{ subject to } \|\beta\| < t \tag{2.8a}$$

$$\hat{\beta}_{lasso} = \arg\min_{\beta} \|y - X\beta\| \text{ subject to } \|\beta\|_1 < t \tag{2.8b}$$

where t is a tuning parameter. There is a one-to-one mapping between t and λ (see Equation (2.6) and Equation (2.7)).

There is a similarity between the ridge regression and Lasso in Equation (2.8). However, the ridge penalty is L^2-*norm* while the Lasso penalty is L^1-*norm*. The ridge regression solution for the problem in Equation (2.8a) is

$$\hat{\beta} = (X^T X + \lambda I)^{-1} X^T y \tag{2.9}$$

where I is the n × n identity matrix. There is no closed form solution to Lasso, since the constraint $\|\beta\|_1$ makes the solution nonlinear in the y. Many effective algorithms as well as quadratic programming are available to solve the Lasso problem [19, 20].

The two most remarkable properties of Lasso have been discussed by Xu et al. [21]. The authors investigated the robustness and sparsity properties provided by the Lasso solution. Robustness is embedded in the regularization scheme through minimization of the worst case residual. Figure 2.1 can be used to explain the sparsity of Lasso. If we consider a linear regression problem with two parameters β_1 and β_2, then the least squares solution is the $\hat{\beta}$, which is shown in the center of the ellipses. Each elliptical contour represents the residual sum of squares or the loss surface. As the distance from $\hat{\beta}$ increases, the loss surface also increases. For this problem, a feasible solution can be obtained by Equation (2.8) where the constraints are $\|\beta_1\|_1 + \|\beta_2\|_1 < t$ for Lasso and $\beta_1^2 + \beta_2^2 < t^2$ for ridge regression. Therefore, the feasible set of solutions is within the regions of these constraints. The regions for these constraints are also drawn in Figure 2.1. The shape of the region is a square for Lasso, whereas it is a circle for ridge regression. Now, the optimal solution will be the point where the contours touch the feasible set of solutions.

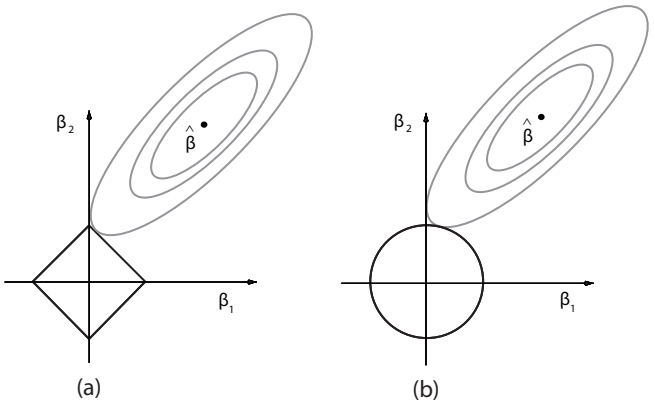
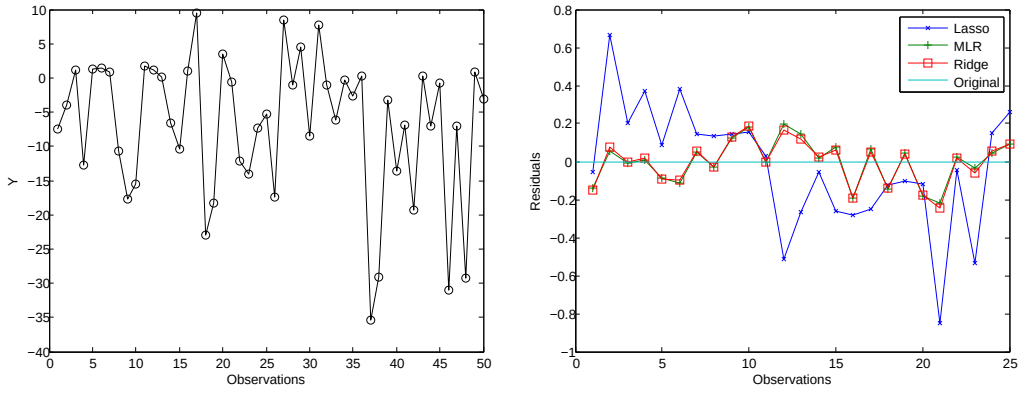


Figure 2.1. *Estimation pictures for (a) Lasso and (b) ridge regression [14].*

For Lasso, the constraint region is a square with corners on the coordinate axes where all but one parameter is exactly zero (see Figure 2.1(a)). Therefore, the contours may touch the squared region either in a corner or on an edge between corners with some of the parameters being exactly zero. On the other hand, there are no corners in the constraint region for ridge regression solution (see Figure 2.1(b)). Hence, a solution with parameters set to exactly zero rarely occurs [14, 16].

Alternatively, the properties of Lasso can be demonstrated by a simple example. Consider a 5-dimensional artificial data $\mathbf{X}$ of 50 samples drawn from exponential distribution with means ranging from 1 to 5. In other words, each column of $\mathbf{X}$ corresponds to an array of random numbers chosen from exponential distribution of i^{th} means where $i = 1 \ldots 5$. Now, we generate the response data $\mathbf{Y}$ such that $Y = X\beta + \varepsilon$ where β is the model parameter with two non-zero components and additive noise ε with $\varepsilon \sim \mathcal{N}(0, 0.1)$. The resultant response is shown in Figure 2.2(a). Now, we are using the first 25 samples of X to build the models. The rest of the samples will be used for prediction. Figure 2.2(b) shows the residuals of predicted responses for different regression methods.



(a) (b)

Figure 2.2. *(a) Example of a response model drawn from the artificial data with 5 components. β = { 0, 2, 0, -3, 0} was chosen as true model parameters.(b) The residual plot of different methods to estimate the model parameters. The prediction performances of MLR and ridge regression are comparatively identical in this example. Therefore, their curves are overlapped.*

Table 2.1. *Estimation of 5 components using different regression methods.*

β	MLR	Ridge regression	Lasso
0	-0.0361	-0.0344	0
2	2.0104	2.0078	1.8844
0	-0.0061	-0.0057	0
-3	-2.9962	-2.9927	-2.9306
0	0.0003	0.0001	0

In Figure 2.2(b), we can see that the residuals between the empirical and estimated responses are similar in both MLR and ridge regression models. It is also evident from the values listed in Table 2.1. On the other hand, Lasso is able to identify and discard the unnecessary components. Although the values (-0.0361, -0.0061, -0.0003) predicted by MLR are closer to zeros, the method cannot ignore or discard the components in many cases. Likewise, we cannot always ignore the nonzero components predicted by ridge regression. Hence, we will exploit these properties in this thesis and take advantages of the Lasso to solve our problem.

2.2 Artificial neural networks

A neural network is a combination of neurons performing operations in parallel. The operation of the network is generally determined by the connections between the neurons. In other words, a neural network is a collection of interconnected nodes which uses mathematical models for information processing. Each node in

the network represents a processing element or neuron. The nodes are connected through links that define the relationship between nodes. The neural network is also known as the *artificial neural network (ANN)* or *simulated neural network (SNN)*. The artificial neural network is simply inspired by the biological neural processing systems in humans or animals.

A simple ANN composed of input and output layers is presented in Figure 2.3. Here, the input layer contains m number of input nodes represented by X_1, X_2, ..., X_m.

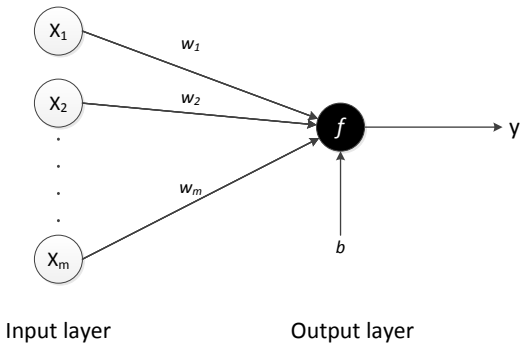


Figure 2.3. *A simple artificial neural network.*

The output layer consists of a real valued *activation function* f. Thus, the resulting node output is defined as

$$y = f(\sum_{j=1}^{m} w_j X_j + b) \tag{2.10}$$

where w_j is the weight of the connection from the input X_j, b is the bias or an intercept, and m is the number of nodes connected. Here, both w_j and b are adjustable scalar parameters. The arrows in Figure 2.3 represent the directions of information flow from node to node. It is the simplest structure consisting of one layer of neurons connected with inputs X_1, X_2, ..., X_m. The weights w_1, w_2, ..., w_m associated with the connections and bias b are trained to produce the desired output. This neuron model is also known as the *perceptron* model. A similar neuron was described by McCulloch and Walter Pitts in the 1940s [22–25].

In ANN, each neuron has an activation function which determines the output to the corresponding neuron to a given input. The most commonly used activation functions are shown in Figure 2.4.

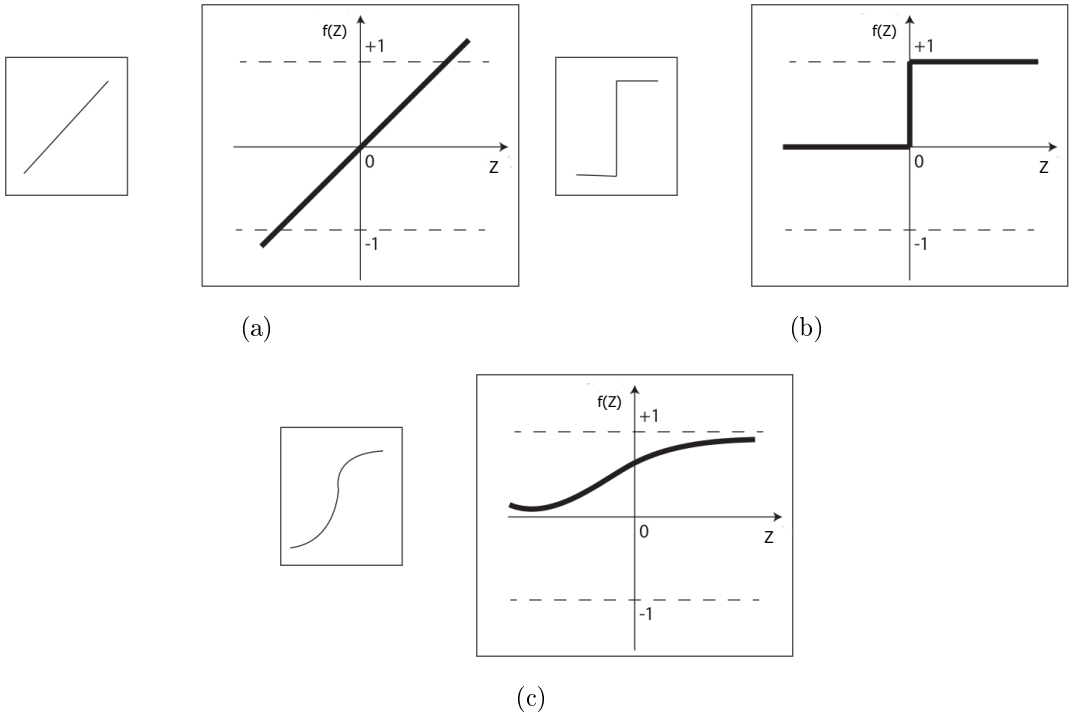


Figure 2.4. *The symbol and graph of various activation functions used in ANN: (a) The linear function, (b) the threshold function and (c) the nonlinear logistic function [26].*

The mapping between inputs and outputs are usually bounded by the activation functions in a given range. The range is either a binary value [-1,1] or a bipolar value [0,1]. In the *linear activation function*, as shown in Figure 2.4(a), the output of the neuron is proportional to the linear combination of multiple weighted inputs with bias. In other words,

$$f(Z) = a\ Z \tag{2.11}$$

and

$$Z = \sum_{j=1}^{m} w_j X_j + b \tag{2.12}$$

where a is a constant value, and Z is a variable defined as the linear combination of multiple weighted inputs with bias. Thus, a perceptron containing the linear activation function is known as the *linear regression model* [27, 28]. The output can also be limited to one of two levels, for instance, either 0 or 1 (see Figure 2.4(b)). In this case, the function is called the *threshold activation function* and the perceptron is known as the *linear discriminant model* [29–31]. In equation form,

$$f(Z) = \begin{cases} 1 & Z \geq a \\ 0 & Z < a \end{cases} \tag{2.13}$$

where a is a constant. A special example of such perceptron model is the *adaline* which has only one output [32]. Another type of activation function shown in Figure 2.4(c) is the *nonlinear logistic function*. Here, the output range is squashed between 0 and 1 for any real value of the input. We can also express the function as

$$f(Z) = \frac{a}{1 + e^{c\ Z\ +\ d}} \tag{2.14}$$

where a, c and d are constant. The perceptron paradigm based on this function is known as the *logistic regression model* [33].

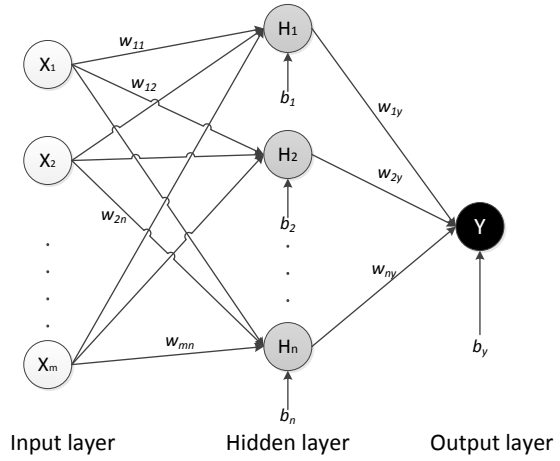


Figure 2.5. *An MLP with one hidden layer.*

Perceptron can be extended with hidden layers, shown in Figure 2.5, which is also known as the *multilayer perceptron* or *MLP*. Typically, the processing elements in the hidden layers are not directly connected to the external world [34]. The number of neurons in hidden layers may differ from the number of neurons in the input and output layers. Usually, hidden layers use nonlinear activation functions such as logistic function. For an MLP with one hidden layer, we can rewrite the basic Equation (2.10) as

$$y = f\left(\sum_{i=1}^{N} f\left(\sum_{j=1}^{m} w_j X_j + b_i\right) + b_y\right) \tag{2.15}$$

where N is the number of neurons in the hidden layer. Being *universal approximator*, MLP is the most widely known and used without any prior knowledge about the input-output relationship [35,36]. Alternatively, MLP can approximate any function to an arbitrary degree of accuracy by increasing the number of neurons in the hidden layer. An MLP with a small number of neurons in the hidden layer can provide a useful alternative to polynomial regression.

Generally, MLP network is trained by *backpropagation* algorithm which computes gradients of the network output with respect to its weights. The gradient vector is obtained recursively by means of chain rule defined as the *delta rule*,

$$w_j = w_{j-1} - \eta \, \delta_j \, x \tag{2.16}$$

where δ_j is the local gradient, x is the input of neuron j, and η is the learning rate. The gradient vector at each node is always computed in the opposite direction of output flow, thus the learning procedure is known as the *backpropagation learning algorithm*. Using this algorithm, the weights in the network are updated epoch by epoch, until a stop criterion is met.

The MLP, as shown in Figure 2.5, is usually interconnected in a feedforward way. That is, the network contains no cycles. Hence, the architecture is also known as the *feedforward network*. In this network, the information moves in from the input nodes, through the hidden nodes to the output node. The behavior of a feedforward network can be divided into two distinct phases: the training or learning phase and the running or activation phase. During running phase, an activation function is applied to each node to produce the desired output. In order to solve nonlinear problems, the logistic activation functions are used in the hidden layers, such as sigmoid function. In the learning phase, the weights and biases are adjusted, thereby changing the performance of the network. Several training algorithms exist in literature for feedforward networks and some of them are discussed in this section.

The basic backpropagation algorithm adjusts the weights and biases in the steepest descent direction (negative of the gradient). Although the function decreases rapidly along the negative of the gradient, this does not ensure fast convergence. An alternative solution can be conjugate gradient algorithms that perform search along conjugate directions, producing faster convergence than steepest descent directions. These algorithms require higher memory storages than the simpler ones. For faster optimization, algorithms based on Newton's method can be used. These are called the *quasi-Newton* (secant) methods. The most popular among these algorithms is the *Broyden, Fletcher, Goldfarb, and Shanno (BFGS) algorithm*. This is a suitable training algorithm with smaller networks, while it requires more computations and storages for larger networks. *Levenberg-Marquardt algorithm* is the fastest method for training a moderate-sized feedforward network. An extension to this algorithm is *Bayesian Regularization* training algorithm which prevents overfitting of the network.

ANN is able to solve the prediction problem efficiently, however, the network may lead to over-parameterization unless carefully designed [14]. Certain issues should be taken into account while training the network:

1. Initial weights with zero values leads to zero derivative making no effect on weights in all iterations, whereas, larger values in weights may lead to inadequate solution. Moreover, weights should be initialized at random values. Otherwise, all weights have the same gradient and they will always be equal.

2. Having too many weights may cause overfitting. This can be minimized by introducing stop criterion such as weight decay.

3. The quality of the final outcome depends on how the inputs have been scaled. Inputs with zero mean and one variance can be considered as a standardized input set.

4. The number of neurons used in each hidden layer has great impact on the network. Using fewer numbers of neurons may be inadequate for capturing nonlinearities in the data.

5. It requires proper knowledge and experiments for finding the reasonable number of hidden layers in a network.

6. The error surface in ANN may possess local minima, thus the training algorithm may entrap in any of those local minima resulting in a poor performance. Several techniques have been discussed in [14] to avoid or overcome such issues.

7. The choice of training algorithm also has a great impact on computational complexity of the network and memory overhead of the system.

In the next chapter, we will describe the algorithms for model assessment and briefly discuss about optimization technique.

3. MODEL ASSESSMENT AND OPTIMIZATION

In this chapter, we are aiming to optimize the response predicted from the obtained models. Before applying an optimization procedure, we emphasize the need to assess the performance of the models that we obtained. For this purpose, we will describe the *cross validation* method. Then we will briefly discuss about a popular global optimization technique - *genetic algorithm*.

3.1 Model assessment

By assessing the performance of a model, we ensure the quality of the chosen model. In other words, we need to find a way that evaluates the prediction methods by the prediction capabilities on unseen dataset. One way of assessing the quality of the obtained models by prediction method is to measure the *loss function* or *cost function*. A loss function δ can be defined as the difference between true values and predicted values of the dependent variable. If true values and predicted values are denoted as y and $\hat{y}$, then a common choice is

$$\delta(y, \hat{y}) = (y - \hat{y})^2 \tag{3.1}$$

Given an experimental dataset, the loss function in Equation (3.1) can estimate the performance of the obtained model which is not only important to future prediction accuracy but also for choosing the best model. Consider the model in Equation (2.1) with n observation samples. We can fit this model, either by Lasso or ANN, and obtain the *residual sum of squares (RSS)* or the sum of squared differences between the true values y and predicted values $\hat{y}$. Thus we can rewrite the Equation (3.1) as

$$RSS = \sum_{i=1}^{n} \delta(y_i, \hat{y}_i) \tag{3.2}$$

Another approach is measuring the degree of linear association or *correlation* between the true values y and predicted values $\hat{y}$. The association can be either positive or negative. In positive correlation, increasing one variable will also increase the other and vice versa. Whereas, negative correlation is the association between

two variables in which one variable increases as the other decreases, and vice versa. The ranges of correlation can vary from +1 to -1. Values close to +1 indicate a high-degree of positive correlation, and values close to -1 indicate a high degree of negative correlation. Values close to zero indicate poor correlation and zero indicates no correlation at all. Several correlation measurements are existed in literature, often denoted ρ or r. The most familiar one is the *Pearson correlation coefficient*, also known as the *coefficient of correlation* [12]. It is obtained by dividing the covariance of the two variables by the product of their standard deviations. We can write

$$\rho = \frac{cov(y, \hat{y})}{\sigma_y \sigma_{\hat{y}}} \tag{3.3}$$

where *cov(y,ŷ)* is the covariance between y and $\hat{y}$, and σ_y and $\sigma_{\hat{y}}$ are the standard deviations of y and $\hat{y}$, respectively. The *cov(y,ŷ)*, σ_y and $\sigma_{\hat{y}}$ can be obtained by

$$\begin{aligned} cov(y, \hat{y}) &= \frac{1}{n}\sum_{i=1}^{n} y_i \hat{y}_i - \sum_{i=1}^{n} y_i \sum_{i=1}^{n} \hat{y}_i \\ \sigma_y &= \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(y_i - \frac{1}{n}\sum_{i=1}^{n} y_i\right)} \\ \sigma_{\hat{y}} &= \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(\hat{y}_i - \frac{1}{n}\sum_{i=1}^{n} \hat{y}_i\right)} \end{aligned} \tag{3.4}$$

Thus, we can rewrite Equation (3.3) in terms of true and predicted values of n observation samples as

$$\rho = \frac{n\sum_{i=1}^{n} y_i \hat{y}_i - \sum_{i=1}^{n} y_i \sum_{i=1}^{n} \hat{y}_i}{\sqrt{n\sum_{i=1}^{n} y_i^2 - \left(\sum_{i=1}^{n} y_i\right)^2}\sqrt{n\sum_{i=1}^{n} \hat{y}_i{}^2 - \left(\sum_{i=1}^{n} \hat{y}_i\right)^2}} \tag{3.5}$$

With an infinite number of samples, the performance of an obtained model may be estimated accurately. However, in real applications, only limited numbers of samples are available. Therefore, we need to split the dataset randomly. Part of the dataset will be used to fit the model, which is called the *training set*. The remaining part of the dataset is used to estimate prediction errors for the model selection, which is defined as the *test set*. Figure 3.1 represents the idea of splitting the dataset. The training set is used to train the model and the test set is used for assessing the performance of the obtained model. Choosing the number of observations in each set is difficult. The dataset might be randomly split into say, 2/3 for the training set and 1/3 for the test set. This method is called the *hold-out method* [37]. This is a suitable method for a large number of training samples and a limited decrease in

the training set does not hinder the quality of the model. However, the performance of the model may significantly vary depending on how the data are split.

Figure 3.1. *Splitting a dataset into the training set and the test set.*

The *hold-out method* is the simplest variation of the *cross validation (CV)* [14]. The most common type of *CV* is the *K-fold cross validation*, also known as the *rotation estimation*, where the dataset is randomly split into K mutually exclusive subsets or the *folds* of approximately equal size. For example, a dataset split into $K = 5$, is shown in Figure 3.2.

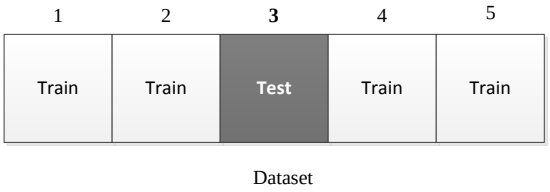


Figure 3.2. *Splitting a dataset into K-folds where $K = 5$.*

In *K-fold cross validation*, a single *fold* is retained as the test set for assessing the model, for instance, the third fold in Figure 3.2. The remaining K - *1* folds are used as training set to fit the model. This process is repeated K times and each of the K folds has been used as the test set exactly once. The K results are then combined or averaged to produce a single estimation of the model. Typical values of K are 5, 10 and 20.

A special case of *K-fold cross validation* is the *Leave-one-out cross validation (LOOCV)*, where $K = n$. In this case, a single observation from the dataset is used as the test set and the remaining part is used for fitting the model. This process is repeated until each of the observation samples has been used as the test set. This method has low bias but can have very high variance.

In this thesis, *CV* is used for both model selection and model assessment. The performances of the models are quite sensitive to the selection parameters such as the λ for Lasso in (2.7) or the number of neurons for ANN in the hidden layers. To estimate the performance with different values of λ for Lasso and different number of neurons for ANN, we use the *K-fold cross validation*.

Using *K-fold cross validation* method, we describe how to select a suitable number of neurons for ANN in the hidden layers (see Algorithm 1). Here, the total number

of neurons is set to 15, since a large number of neurons in the hidden layer may increase the complexity of the ANN model as well as may cause overfitting. Next, we split the dataset into K folds for m^{th} number of neurons. Then, at each fold, we retain the F^{th} fold for testing and remaining folds for training the network for m^{th} number of neurons. The coefficient of correlation ρ in Equation (3.5) is computed for F^{th} fold. This process is repeated until each fold is evaluated for m^{th} number of neurons and the results are averaged. The number of neurons which yields the maximum of the average ρ is selected by this Algorithm 1.

$totalNumberOfNeurons \leftarrow 15$
for $m = 1$ `to` $totalNumberOfNeurons$ **do**
  `Split the` *samples* `into` K `folds`
  for $F = 1$ `to` K `folds` **do**
    $testSet \leftarrow$ `select` *samples* `from` F `fold`
    $trainingSet \leftarrow$ `select` *samples* `from all other folds except` F
    `Train the network with` $trainingSet$ `for` m `number of neurons`
    `Simulate the trained network with` $testSet$
    $correlation \leftarrow$ `compute` ρ `using Equation` (3.5) `for predicted and true responses of` F^{th} `fold`
  end for
  $averageCorr \leftarrow$ `compute the average of` $correlation$ `for` m^{th} `number of neurons`
end for
`Select the number of neurons with maximum` $averageCorr$

Algorithm 1. *Selecting the number of neurons*

In order to assess the performance of the models appropriately, we use the $LOOCV$ for each model to estimate the prediction error. For each observation sample i, we create a *testSet* for i^{th} sample and a *trainingSet* for the remaining samples. The models are constructed using the *trainingSet* for each of the prediction method described in Chapter 2. Then we predicted the response of the i^{th} sample. The steps are summarized in Algorithm 2. For each model, the algorithm computes the coefficient of correlation ρ described in Equation (3.5).

$n \leftarrow totalNumberOfSamples$
for $i = 1$ `to` n **do**
  $testSet \leftarrow ithSample$
  $trainingSet \leftarrow allSamples \neq i$
  `Construct the model with` *trainingSet* `using the selected method`
  `Predict the response with` *testSet* `using the constructed model`
  `Store the predicted response`
end for
`Compute` ρ `using Equation` (3.5) `for predicted and actual responses`

Algorithm 2. *Leave-one-out cross validation*

In the next section, we will discuss about the optimization technique which is applied for further improvement.

3.2 Optimization

Optimization is a process of selecting the best alternative from an available set of alternatives. Therefore, it requires defining a set of potential alternatives and determining the best one. In general, the main objective of the optimization problem concerns with the maximization or minimization of a real function defined by the problem-specific domain. For instance, in this thesis we will maximize the predicted response. Depending on the nature of the function, optimization problems can be divided into discrete and continuous problems. Discrete problems are restricted to discrete variables, such as integer. In discrete problems, finding an optimal solution is a trivial procedure, since a unique optimum always exists. On the other hand, continuous problems consist of real-valued variables and the search space is usually infinite [38]. Several local and global techniques are available for solving the continuous nonlinear optimization problems for example, gradient descent, genetic algorithms and tabu search [39]. In this thesis, our particular focus is on the genetic algorithms.

Genetic algorithm (GA) is one of the most popular global optimization methods. This algorithm is motivated by so-called *nature's wisdom*: the concepts of natural selection and evaluation processes [40]. The optimization methods are associated with minimization (or maximization) of a given objective function. GA provides a framework [41] for solving linear and nonlinear problems by searching through a space of potential solutions. The major components in the framework include encoding schemes, fitness evaluation, selection of parents, crossover, and mutation which will be discussed later in this chapter.

3.2.1 Framework of genetic algorithm

The terminologies in GA are adapted from biological processes in natural system. A *chromosome* consists of strings of DNA in which organisms' genotype is stored. Each chromosome can be partitioned into *genes* which are located in a particular *locus* on that chromosome. The locus is also known as the *crossover position* where the reproduction of a new chromosome takes place. The organism that holds the new chromosome is called an *offspring*. During reproduction, *recombination (crossover)* occurs by combining the characteristics of two or more parent chromosomes to form an offspring. The alteration of single gene may occur randomly, which is known as the *mutation*, throughout the recombination process. However, mutation is relatively a rare process, caused either by error during replication of parents' genes or irrecoverable damage to an element of a chromosome. A set of new offspring forms a new *generation* and the total number of offspring at a particular time is known as the *population*. Each member of the population is evaluated for *fitness* in each generation and members with higher fitness values participate in developing a next generation.

Likewise, chromosomes in a GA population are a string of bits designed by specific encoding scheme. Each bit represents a gene, having two possible states: 0 and 1. Each chromosome refers to a point in search (solution) space of candidate solutions. All points in the search space are associated with a fitness value, which is typically an objective function evaluated at the corresponding points in the solution space. Examples of such objective functions can be, for instance, the sum of squares error between predicted and experimental response.

In each generation, chromosomes in the current population are evaluated for fitness. Members with higher fitness values are more likely to participate in reproduction using *genetic operators*, for instance, crossover and mutation. As a result, a new population is constructed from a set of newly produced chromosomes and replaces the current population. This new population then participates for genetic operations in the next generation. After a number of generations, the population of GA contains members with better fitness values. The basic steps of GA are summarized in Algorithm 3. In GA, generations are iterated until a desired termination criterion has been satisfied. Termination criteria can be, for instance, a predefined number of generations or reaching the minimum fitness limit.

```
initialize population
while termination criteria have not been met do
  evaluate population
  select chromosomes for reproduction
  perform crossover and mutation
  accept new generation
end while
```

Algorithm 3. *Basic genetic algorithm*

3.2.2 Major components in genetic algorithm framework

GA framework requires the determination of five fundamental components: encoding scheme, evaluation of fitness, selection of parents, crossover, and mutation. The rest of this section will briefly discuss these components.

Encoding schemes

Encoding schemes define the representation of the information contained by a chromosome in the search space. For instance, using binary coding, a 2-dimensional point (9, 5) can be transformed in the GA framework where each coordinate will be represented with 8 binary bits. Thus, the result will be (00001001, 00000101). The operations, such as crossover and mutations performed on populations, are designed based on the encoding schemes. Binary coding is applied in the original framework of GA [40]. The basic encoding scheme has also been extended to gray coding and diploid binary encoding scheme. Other encoding schemes such as value encoding, tree encoding can also be used [40, 42–44].

Evaluation of fitness

After creating a generation, each chromosome in the current population are evaluated for fitness using an objective function. The purpose of the objective function is to provide an assessment of the performance of chromosomes in the problem domain. A chromosome i in the population can be thought of as a point in search space associated with a fitness value f_i. A *fitness landscape* includes all possible solutions along with their fitness values in the search space. Figure 3.3 is an example of fitness landscape with *hills*, *valleys*, and *peaks*. The process of evaluation allows members of the populations to move across the landscape, particularly towards peaks. This movement is defined by the objective function of the problem domain.

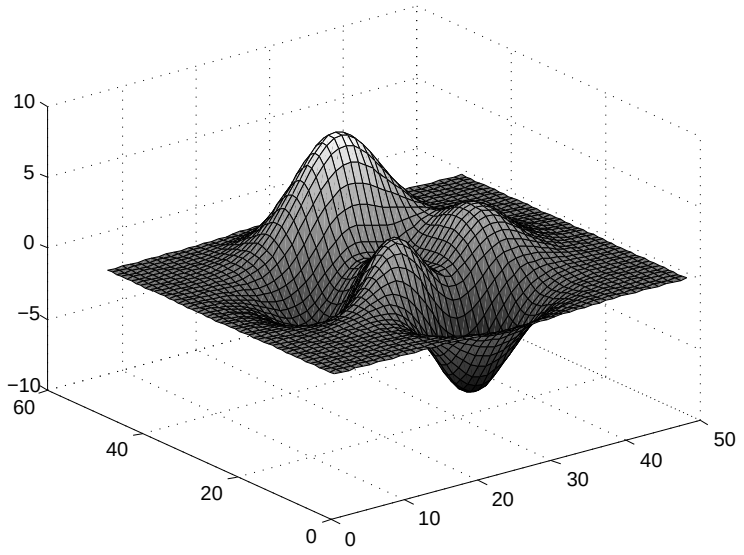


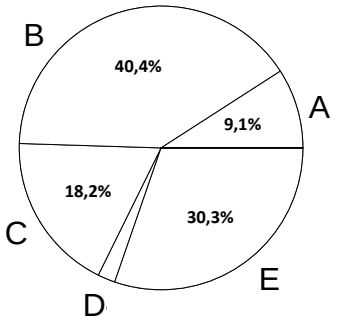
Figure 3.3. *An example of fitness landscape.*

Selection of parents

For reproduction, chromosomes are selected from the current population according to their corresponding fitness values. The selection operation determines which chromosomes will be considered for creating offspring for the next generation. In general, chromosomes with higher fitness values are chosen. Each chromosome is assigned with a probability proportional to its fitness value. This assignment can be easily implemented using a simple method known as the *roulette wheel method* [43]. In roulette wheel method, a slice of the *roulette wheel* is assigned to each member in the population where the size of the slice being proportional to the selection probability of that member's fitness. The wheel is then, spins N times to select N number of chromosomes. The selection probability for i^{th} member is equal to its fitness f_i divided by the total fitness of all members in the current population, that is $f_i / \sum_{k=1}^{n} f_k$, where n is the size of the current population. Figure 3.4 shows the probability of being selected for each chromosome in a population of five. Chromosome B dominates the graph wheel because its fitness value is significantly greater (40.4%) than those of the other four. As a result, chromosome B is much more likely to be selected as a parent, whereas chromosome D is less likely to be chosen (2%).

Chromosome	fitness f_i	Probability (%)
A	45	9.1
B	200	40.4
C	90	18.2
D	10	2
E	150	30.3

(a)



(b)

Figure 3.4. *An example of selection probability assigned by the roulette wheel method. (a) The list of fitness values and probabilities (in %) of five chromosomes in a population. (b) A pie chart of probability for the chromosomes being selected.*

Another approach is the *stochastic universal sampling* (SUS) [45], which is similar to the roulette wheel method. However, instead of selecting a chromosome according to the assigned probability, the method selects the chromosomes at evenly spaced intervals. This allows the weaker (lower fitness values) chromosomes a chance to participate in reproduction, thereby reducing the dominance of highly fitted chromosomes.

For the selection methods, a chromosome can be selected more than once. If highly fitness chromosomes are always selected in reproduction, then suboptimal chromosomes may dominate the population. As a result, the ability of the algorithm to find the global optimum may reduce. Instead, if the selection criterion is diversified, the convergence of the model to global optimum may be too slow. Various techniques can be applied for balancing the selection criterion either by increasing emphasis on favoring highly fitness chromosomes or by allowing weaker fitness members to survive. For example, De Jong [46] developed a method called the *elitism*, which retains a certain number of best chromosomes from one generation to the next. This method considerably improves the performance of GA [41,47]. Another alternative method is the *rank selection* [48], where members are ranked according to their fitness and the selection depends on the ranks rather than absolute fitness values. The purpose of rank selection is to prevent convergence to a local optimum. Other popular methods are *sigma scaling* [49], *boltzmann selection* [50], and *tournament ranking* [51].

Crossover

The crossover operation is applied to the selected pairs of chromosomes to produce offspring for the next generation. This is usually done with a probability equal to a given *crossover rate* (p_c). In this operation, the crossover positions in parents'

genes are chosen randomly and part of the parents' chromosomes are interchanged. The purpose of the crossover is to generate new offspring which may retain good characteristics from the previous generation. Researchers have implemented many crossover methods [43, 52] and some of them have been described in this section.

One-point crossover is the most basic crossover operation where the position is chosen randomly and the subsequences of the parents' chromosomes are interchanged beyond that position. In *two-point crossover*, two positions are chosen randomly and the subsequences of the parent chromosomes between these two positions are swapped. Similarly, the concept can be defined for *k-point crossover*. Examples for one-point and two-point crossover operations are shown in Figure 3.5. Another common crossover operation is the *uniform crossover*, where each gene is exchanged between parent chromosomes with a *swapping probability*. This probability is typically set to 50% [53, 54]. Figure 3.5 shows an example of uniform crossover.

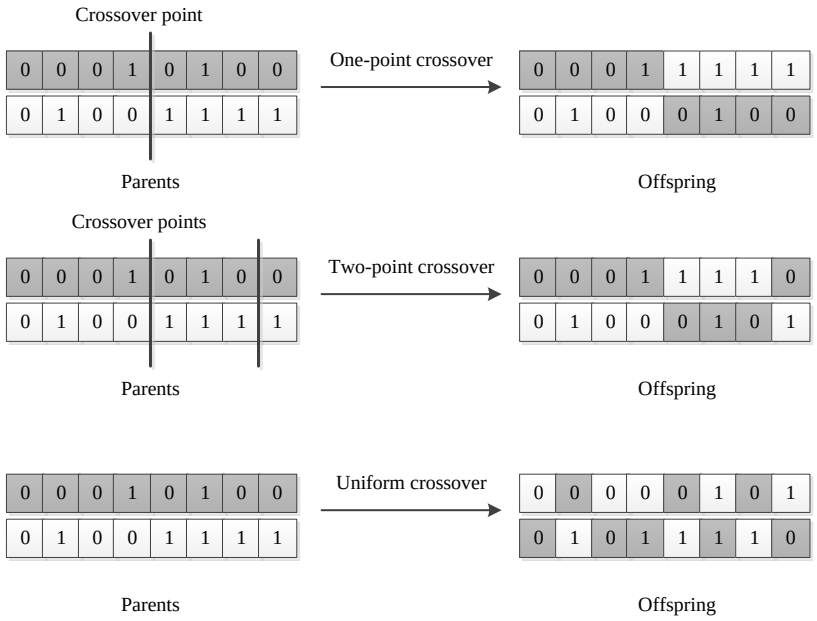


Figure 3.5. *One point, two point and uniform crossover.*

Mutation

A simple way to implement mutation is to alter the bits of an offspring randomly with a very low probability, known as the *mutation rate* (p_m). Mutation introduces diversity to the population as well as ensures the possibility of exploring the entire search space. An example of mutation is illustrated in Figure 3.6. Usually, mutation rate is kept very low, typically between 0.001∼0.05, thus, good offspring are not lost. Thereby, prevents the population from converging too quickly to a local optimum.

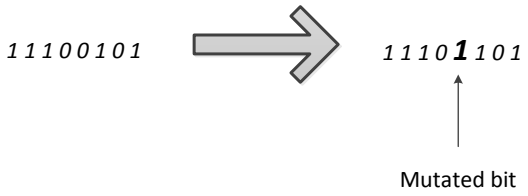


Figure 3.6. *An example of mutation. In this operation, the 5^{th} bit is altered from 0 to 1.*

In the next section, we will illustrate two simple examples to understand these concepts of GA.

3.2.3 Examples of genetic algorithm

Consider a normal distribution function of x

$$f(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right) \tag{3.6}$$

where μ is the mean and σ is the standard deviation. We would like to find out the maximum value of *f(x)* with $\mu = 8$ and $\sigma = 2$. Figure 3.7 displays Equation (3.6) where x taking the values from 0 to 15.

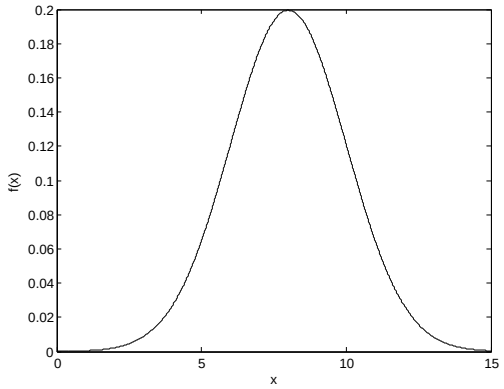


Figure 3.7. *Finding the maximum value of the normal distribution function with $\mu = 8$ and $\sigma = 2$ using GA.*

Using four digit binary encoding scheme, we can represent the values of x in the range from 0000 to 1111. We also assume that the crossover rate p_c and the mutation rate p_m are 0.75 and 0.002, respectively. With a population size of 4, we choose four chromosomes randomly from the set $0000 - 1111$. They are, for instance, 0101 (5), 1001 (9), 1100(12), and 1111 (15).

At first iteration, we compute the fitness function *f(x)* for all chromosomes in the current population which are listed in Table 3.1. For selection of the parents, roulette wheel approach can be used. The total fitness of all the chromosomes in the current

population is $\sum_{k=1}^{4} f_k = 0.268223$. Hence, the selection probability of chromosome 5, for instance, is $0.064759/0.268223 = 0.241436$. Similarly, the selection probability of other chromosomes are calculated which are shown in Table 3.1. According to the fitness values, chromosome 9 has the highest probability of being selected. Since chromosome 5 and chromosome 9 have higher selection probability in the current population, we can assume that they are selected as the first pair of parents.

Table 3.1. *Fitness and Selection probability of randomly chosen chromosomes in the first iteration of GA.*

Chromosome	Binary value	Fitness *f(x)*	Selection probability
5	0101	0.064759	0.241436
9	1001	0.176033	0.656292
12	1100	0.026995	0.100646
15	1111	0.000436	0.001627

If one-point crossover takes place between 5 (0101) and 9 (1001) at second position, then each parent chromosome will be partitioned into two parts at the crossover point. That is, 5 (0101) will be segmented into 0 and 101, while 9 (1001) into 1 and 001. Now, each child chromosome will receive one segment from each of the parents. Thus, the two new chromosomes will be 1 (0001) and 13 (1101). Additionally, chromosome 9 and chromosome 12 are randomly chosen as second pair of parents by the roulette wheel method. In this case, we assume that no crossover has taken place. Therefore, the members of new population will be 1, 13, 9 and 12, which will replace the current population. The iteration will be continued until the stop criterion has been satisfied.

The progress of GA across generations can be viewed in Figure 3.8. This plot illustrates the best and average values of the fitness function across 50 generations. After thirty fifth generations, the population starts to converge to peak containing the maximum.

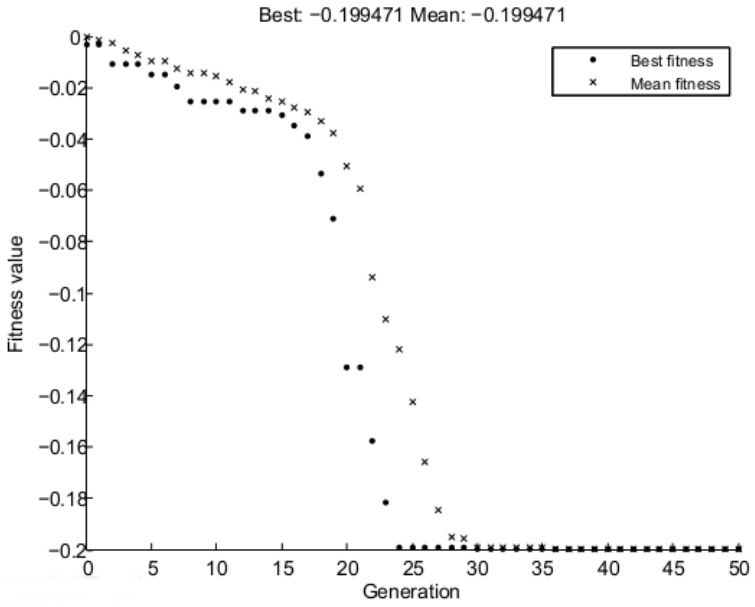


Figure 3.8. *The performance of GA across 50 generations.*

Another example of finding maximum value of a sinusoidal function of x defined as

$$f(x) = \frac{sin(10x)^2}{x+1} \tag{3.7}$$

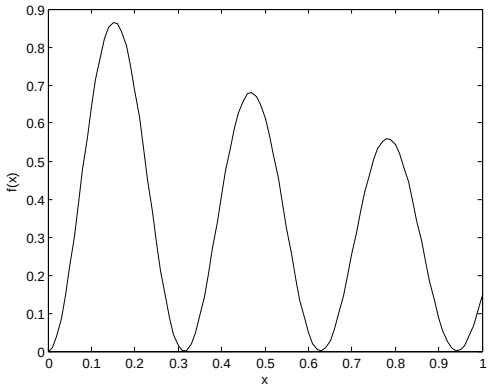


Figure 3.9. *Another example of GA where the function has one global minimum and two local minima. The goal is to find the global minimum using GA algorithm.*

Figure 3.9 illustrates Equation (3.7) for the values of x ranging between 0 and 1. In this example, the GA algorithm uses a population size of 20 to find the maximum. Figure 3.10 shows the population after 1, 10, 15, and 35 generations with the locations of chromosomes denoted by circle. In the first generation, the chromosomes in the population are scattered throughout the curve, as shown in Figure 3.10(a). As the number of generations increases, the chromosomes in the population get closer together and approach the global maximum point in the curve

(see Figure 3.10(b), Figure 3.10(c) and Figure 3.10(d)).

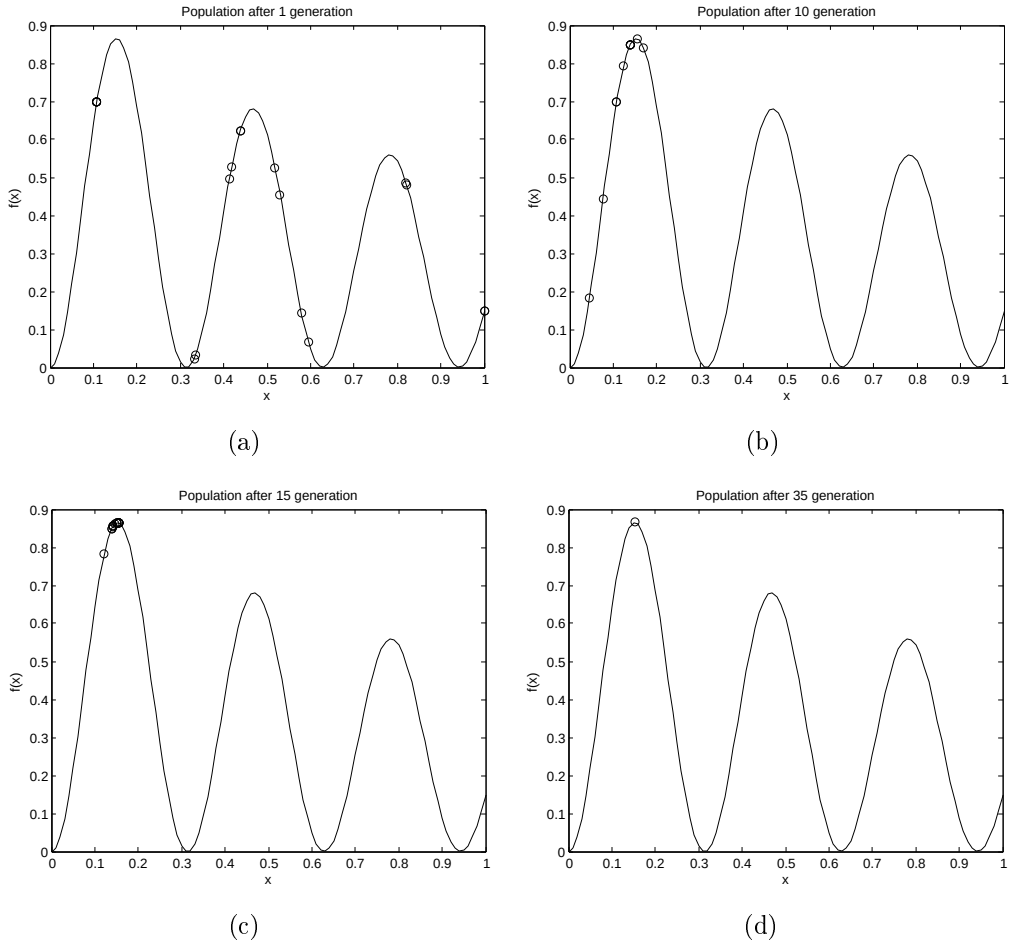


Figure 3.10. *The movement of chromosomes towards the global optimum.*

In the next chapter, we will discuss in details about the experimental data that has been analyzed for this work.

4. CASE STUDY MATERIALS

In this thesis, we examined a dataset obtained in a study on bioconversion of crude glycerol to hydrogen (H_2) by microbes [55]. The authors used a modified HM100 medium containing crude glycerol as enrichment and growth medium, and optimized the media components during bioprocess. The media components in HM100 and their corresponding concentrations are listed in Table 4.1. To enhance H_2 production, initial pH of 6.5 and cultivation temperature at 40°C were chosen for the medium.

Table 4.1. *List of components and their corresponding concentrations in modified HM100 medium.*

Components	Concentration
NH_4Cl	1.0 g/L
K_2HPO_4	0.3 g/L
KH_2PO_4	0.3 g/L
$MgCl_2.6H_2O$	2.0 g/L
KCl	4.0 g/L
$C_2H_3NaO_2.3H_2O$	1.0 g/L
$C_4H_{11}NO_3.HCl$	2.0 g/L
$Na_2S_2O_4$	0.5 g/L
$C_{12}H_7NO_4$	0.002 g/L

Rahul et al. [55] applied statistical experiments for screening and identifying the important medium components in the optimization procedure. First, Plackett-Burman [56] design was applied to study the significance of NH_4Cl, K_2HPO_4, KH_2PO_4, $MgCl_2.6H_2O$ and KCl in production of H_2. Table 4.2 presents the experimental design and the results of Plackett-Burman design. The concentrations of the selected components and the corresponding yield responses were measured in g/L and mol-H_2/mol-glycerol$_{consumed}$, respectively. The rest of the medium components were set in the concentrations of 1.0 g/L, 2.0 g/L, 0.5 g/L and 0.002 g/L for $C_2H_3NaO_2.3H_2O$, $C_4H_{11}NO_3.HCl$, $Na_2S_2O_4$ and $C_{12}H_7NO_4$, respectively. The components NH_4Cl, K_2HPO_4 and KH_2PO_4 were selected for subsequent experiments keeping $MgCl_2.6H_2O$ and KCl in the lowest reasonable concentrations.

Table 4.2. *Experimental design and the results of Plackett-Burman design*

Run	NH_4Cl g/L	K_2HPO_4 g/L	KH_2PO_4 g/L	$MgCl_2.6H_2O$ g/L	KCl g/L	Yield mol-H_2/mol-glycerol$_{consumed}$
1	2.00	1.00	1.00	4.00	8.00	0.37
2	0.50	1.00	0.10	4.00	1.00	0.32
3	2.00	0.10	0.10	4.00	8.00	0.15
4	0.50	0.10	1.00	4.00	1.00	0.32
5	2.00	1.00	1.00	1.00	1.00	0.72
6	0.50	1.00	0.10	1.00	8.00	0.35
7	2.00	0.10	0.10	1.00	1.00	0.57
8	0.50	0.10	1.00	1.00	8.00	0.35

Next, the path of steepest ascent was employed to determine the direction of the selected components. Table 4.3 illustrates the results of the steepest ascent experiment. The selected components were further optimized using Box-Behnken design [57] and ridge analysis [58]. After the ridge analysis experiment, the components NH_4Cl and KH_2PO_4 were considered for optimization. Hence, central composite face centered cube design [59] was adopted for maximizing the H_2 yield by identifying the optimal concentrations of these two components. Thus, maximal H_2 yield was predicted to be 1.41 mol-H_2/mol-glycerol$_{consumed}$. Rahul et al. also confirmed the maximum observable H_2 yield to be 1.42±0.15 mol-H_2/mol-glycerol$_{consumed}$ at the optimum settings of medium components achieved by the design of experiments. The experimental designs and the results of Box-Behnken design, ridge analysis, and central composite face centered cube design are presented in Table 4.4, Table 4.5, and Table 4.6, respectively. A more complete discussion of these designs and related topics can be found in the book by Montgomery [7].

Table 4.3. *Experimental design and the results of the steepest ascent*

Run	NH_4Cl g/L	K_2HPO_4 g/L	KH_2PO_4 g/L	$MgCl_2.6H_2O$ g/L	KCl g/L	Yield mol-H_2/mol-glycerol$_{consumed}$
1	1.25	0.60	0.60	1.00	1.00	0.43
2	2.00	1.60	1.60	1.00	1.00	1.04
3	2.80	2.60	2.70	1.00	1.00	0.87
4	3.60	3.70	3.80	1.00	1.00	0.49
5	4.40	4.70	4.80	1.00	1.00	0.47
6	5.20	5.60	5.90	1.00	1.00	0.44

Table 4.4. *Experimental design and the results of Box-Behnken design*

Run	NH_4Cl g/L	K_2HPO_4 g/L	KH_2PO_4 g/L	$MgCl_2.6H_2O$ g/L	KCl g/L	Yield mol-H_2/mol-glycerol$_{consumed}$
1	0.50	1.10	1.60	1.00	1.00	0.56
2	3.50	1.10	1.60	1.00	1.00	0.90
3	0.50	2.10	1.60	1.00	1.00	0.57
4	3.50	2.10	1.60	1.00	1.00	1.09
5	0.50	1.60	1.10	1.00	1.00	0.83
6	3.50	1.60	1.10	1.00	1.00	1.14
7	0.50	1.60	2.10	1.00	1.00	0.77
8	3.50	1.60	2.10	1.00	1.00	1.35
9	2.00	1.10	1.10	1.00	1.00	0.76
10	2.00	2.10	1.10	1.00	1.00	0.94
11	2.00	1.10	2.10	1.00	1.00	0.91
12	2.00	2.10	2.10	1.00	1.00	0.89
13	2.00	1.60	1.60	1.00	1.00	1.04
14	2.00	1.60	1.60	1.00	1.00	1.05
15	2.00	1.60	1.60	1.00	1.00	1.07

Table 4.5. *Experimental design and the results of the ridge analysis*

Run	NH_4Cl g/L	K_2HPO_4 g/L	KH_2PO_4 g/L	$MgCl_2.6H_2O$ g/L	KCl g/L	Yield mol-H_2/mol-glycerol$_{consumed}$
1	2.00	1.60	1.60	1.00	1.00	1.03
2	3.33	1.64	1.96	1.00	1.00	1.09
3	4.05	1.616	2.52	1.00	1.00	1.39
4	4.63	1.58	3.10	1.00	1.00	1.19
5	5.17	1.55	3.64	1.00	1.00	1.01
6	5.70	1.50	4.20	1.00	1.00	0.81

Table 4.6. *Experimental design and the results of central composite face centered cube design*

Run	NH_4Cl g/L	K_2HPO_4 g/L	KH_2PO_4 g/L	$MgCl_2.6H_2O$ g/L	KCl g/L	Yield mol-H_2/mol-glycerol$_{consumed}$
1	2.95	1.52	1.60	1.00	1.00	0.56
2	5.15	1.52	1.60	1.00	1.00	0.90
3	2.95	3.52	1.60	1.00	1.00	0.57
4	5.15	3.52	1.60	1.00	1.00	1.09
5	2.95	2.52	1.60	1.00	1.00	0.83
6	5.15	2.52	1.60	1.00	1.00	1.14
7	4.05	1.52	1.60	1.00	1.00	0.77
8	4.05	3.52	1.60	1.00	1.00	1.35
9	4.05	2.52	1.60	1.00	1.00	0.76
10	4.05	2.52	1.60	1.00	1.00	0.94
11	4.05	2.52	1.60	1.00	1.00	0.91
12	4.05	2.52	1.60	1.00	1.00	0.89
13	4.05	2.52	1.60	1.00	1.00	1.04

In this thesis work, the data from Table 4.2 to Table 4.6 are aggregated into a single dataset that contains five variables and 48 samples. A nonlinear transformed dataset is also prepared to inspect the relationships between medium components. This transformation includes linear, cross product and squared values of the experimental dataset.

5. EXPERIMENTS AND RESULTS

This chapter describes the necessary steps to evaluate the prediction accuracy of the studied methods. First, models are developed for H_2 production dataset presented in Chapter 4. We then examine the performance of each model to further obtain the optimum H_2 yield. Besides, the results are also briefly discussed.

5.1 Yield prediction

In this section, we will use the H_2 production dataset to build the models described in Chapter 2 and thereby estimate the unknown parameters for the models and identify the significant variables.

5.1.1 Multiple linear regression

Linear equation for the variables $x_1, \ldots, x_5$ can be expressed using Equation (2.2) as follows:

$$y = \beta_0 + \sum_{i=1}^{5} x_i \beta_i \tag{5.1}$$

where β_0 is the intercept and β_i is the coefficient of the variable x_i. By applying Matlab standard function `regress` on the experimental dataset, we obtained the following linear MLR model for yield prediction:

$$y = 1.0306 + 0.1603x_1 - 0.0019x_2 - 0.1676x_3 - 0.1462x_4 - 0.0584x_5 \tag{5.2}$$

where y is the predicted yield; x_1, x_2, x_3, x_4 *and* x_5 are the actual values of NH_4Cl, K_2HPO_4, KH_2PO_4, $MgCl_2.6H_2O$ and KCl, respectively.

Using the Matlab software, the contour curves are drawn for the linear MLR model (see Figure 5.1). Here, each plot represents the effect of two variables on the yield response while other remaining variables are set at their corresponding average levels. The shape of each plot indicates whether the interactions between independent variables are significant or not. Since the model is linear, these contours are parallel straight lines indicating only the main effects. The diagonal plots are left blank since each variable does not have any impact on itself. The grid points in each plot were chosen in the ranges between minimum and maximum values of the corresponding variables. The interval between the grid points was kept to value 0.2.

As shown in Figure 5.1, the darker red indicates the maximum yield response. On the contrary, the minimum yield response is expressed with the darker blue.

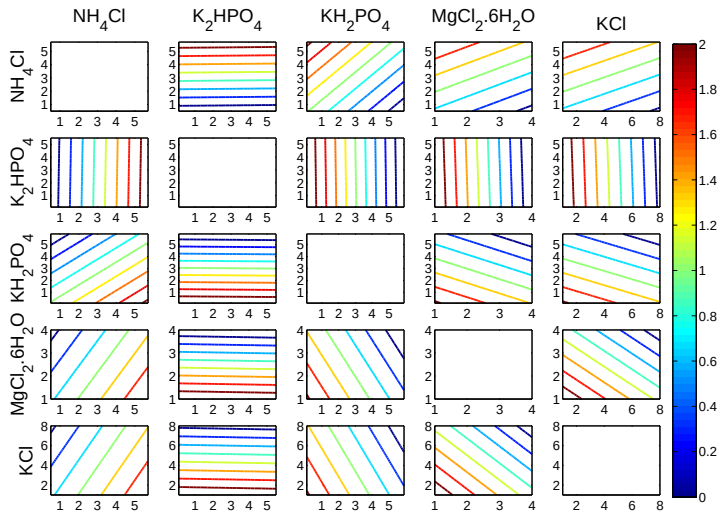


Figure 5.1. *The effects of the yield for pairwise components using the linear MLR model. The yield responses are presented by different colors according to the colorbar. The plots in the diagonal (that is, components are plotted against themselves) are left empty.*

Consider the interactions between K_2HPO_4 and other components in Figure 5.1. The shape of the contour plots between K_2HPO_4 and other components clearly indicates that there were less or no interactions. It is also evident from the lower coefficient value (-0.0019) (see Equation (5.2)). On the other hand, the interactions between NH_4Cl and KH_2PO_4, $MgCl_2.6H_2O$ and KCl have significant effect on the yield responses. Hence, the results indicate, for instance, the yield responses are maximum for the higher concentration values of NH_4Cl and the lower concentration values of KH_2PO_4, $MgCl_2.6H_2O$ and KCl. The interactions among other variables (KH_2PO_4 - $MgCl_2.6H_2O$, KH_2PO_4 - KCl, and $MgCl_2.6H_2O$ - KCl) have negative effect. That is, if we want to maximize the yield, these components should be at the low level.

Quadratic model for the variables $x_1, \ldots, x_5$ can be expressed as following:

$$y = \beta_0 + \sum_{i=1}^{5} x_i\beta_i + \sum_{i=1}^{5} x_i^2\beta_{ii} + \sum_{i<j}\sum x_i x_j \beta_{ij} \tag{5.3}$$

where β_0 is the intercept and β_is are the linear coefficients, β_{ij}s are the pairwise products coefficients, and β_{ii}s are the squared coefficients of the corresponding variables. The transformed dataset was used to implement the quadratic MLR model. By applying Matlab standard function `regress` on the experimental transformed

dataset, we obtained the following quadratic model for yield prediction:

$$\begin{aligned} y = {} & 0.4541 + 0.0639x_1 + 0.3608x_2 + 0.0215x_3 - 0.0586x_1x_2 - 0.0490x_1x_3 \\ & + 0.09x_2x_3 - 0.0692x_2x_4 - 0.0088x_2x_5 + 0.0209x_3x_5 - 0.0001x_4x_5 \\ & - 0.0178x_1^2 - 0.1405x_2^2 - 0.1064x_3^2 - 0.0085x_4^2 - 0.0041x_5^2 \end{aligned} \tag{5.4}$$

where y is the predicted yield; x_1, x_2, x_3, x_4 *and* x_5 are the actual values of NH_4Cl, K_2HPO_4, KH_2PO_4, $MgCl_2.6H_2O$ and KCl, respectively. In Figure 5.2, the quadratic MLR model has been illustrated by the contour plots. The curves in the contours indicate that the model contains interaction and quadratic terms. The interactions between NH_4Cl and KH_2PO_4, $MgCl_2.6H_2O$ and KCl have similar significant effect on the yield responses as in Figure 5.1. However, the graph shows that the interactions among other variables are quite unpredictable.

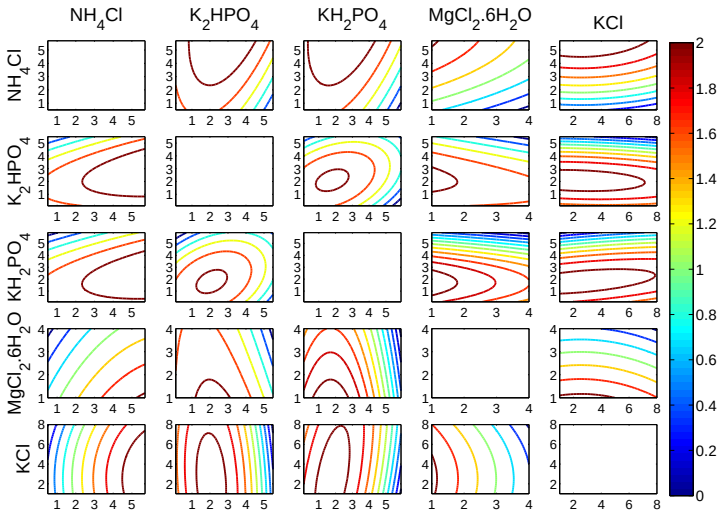


Figure 5.2. *The effects of the yield for pairwise components using the quadratic MLR model. The yield responses are presented by different colors according to the colorbar. The plots in the diagonal (that is, components are plotted against themselves) are left empty.*

5.1.2 Lasso

The linear and quadratic model with Lasso regularization was implemented using the *glmnet package* [60] for Matlab. The package provides `glmnet`, `cvglmnet` and `glmnetCoef` functions for model fitting, validating and identifying the unknown parameters, respectively. The function `glmnet` generates a sequence of models for different values of the regularization parameter λ. The obtained models are assessed using *10-fold cross validation* by `cvglmnet` and the model with the smallest λ is selected. The function `glmnetCoef` determines the unknown coefficients of the model for selected λ.

By applying this Lasso package on the experimental dataset, we obtained the linear model in Equation (5.5). For the selection of model parameter λ, we performed *10-fold cross validation* by `cvglmnet` on the experimental dataset. In this case, the smallest value of λ was 0.0072564.

$$y = 1.0107 + 0.1501x_1 - 0.1506x_3 - 0.1384x_4 - 0.0550x_5 \tag{5.5}$$

where y is the predicted yield; x_1, x_2, x_3, x_4 *and* x_5 are the actual values of NH_4Cl, K_2HPO_4, KH_2PO_4, $MgCl_2.6H_2O$ and KCl, respectively. From Equation (5.5), we can clearly see that the sparsity property of lasso is able to ignore the unnecessary component K_2HPO_4. Using the same procedure on the experimental transformed dataset, we obtained the following quadratic model for yield prediction:

$$y = 0.8162 + 0.1444x_1 - 0.1122x_4 - 0.0269x_5 - 0.0277x_3^2 - 0.0019x_5^2 \tag{5.6}$$

where y is the predicted yield; x_1, x_2, x_3, x_4 *and* x_5 are the actual values of NH_4Cl, K_2HPO_4, KH_2PO_4, $MgCl_2.6H_2O$ and KCl, respectively. *10-fold cross validation* by `cvglmnet` was performed on transformed dataset for the selection of model parameter λ. The minimum value of λ in this case was 0.018398.

Using the Matlab software, the contour curves described by the linear and quadratic models are shown in Figure 5.3 and Figure 5.4, respectively. Here, each plot represents the effect of two variables while the other remaining variables are at their corresponding average level. The shape of each plot indicates whether the interactions between independent variables are significant or not. The parallel straight lines also indicate the linear model contains only the main effects. In Figure 5.4, since the quadratic Lasso model also has the interaction and quadratic terms, the contour plots contain curves. We obtained similar conclusions in the interaction graphs in Figure 5.3 and Figure 5.4 as in the linear MLR model.

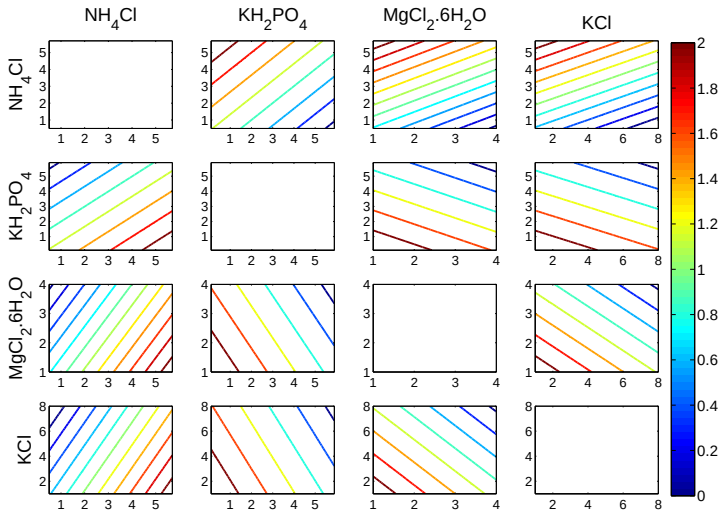


Figure 5.3. *The effects of the yield for pairwise components using the linear Lasso model. The yield responses are presented by different colors according to the colorbar. The plots in the diagonal (that is, components are plotted against themselves) are left empty.*

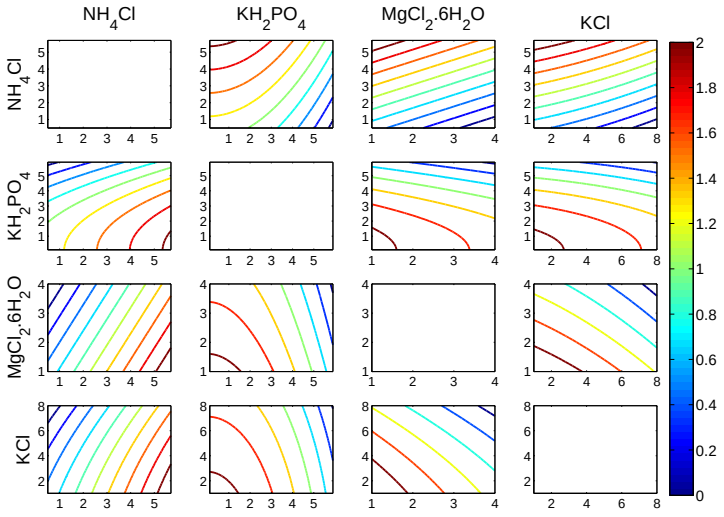


Figure 5.4. *The effects of the yield for pairwise components using the quadratic Lasso model. The yield responses are presented by different colors according to the colorbar. The plots in the diagonal (that is, components are plotted against themselves) are left empty.*

5.1.3 Artificial neural networks

In this section, we have used the Neural Network Toolbox [26] to create two feed-forward networks using the built-in function `newff`. The first one was two-layer network composed of one hidden layer and one output layer. The latter one was three-layer network composed of two hidden layers and one output layer. In the hidden layers, we used the nonlinear logistic activation function *logsig* as shown in

Figure 2.4(c). This would allow the network to learn the nonlinear relationships between inputs and outputs. The linear activation function *purelin* in the output layer would be beneficial in finding the linear approximation to the nonlinear function. The sufficient number of neurons in the hidden layers could be found using *K-fold cross validation method* described in Algorithm 1, where K = 5. Instead of testing the ANN networks with an arbitrary number of neurons in the hidden layers, the algorithm examined the possible number of neurons iteratively. The value with the highest coefficient of correlation was chosen for further investigation. For this purpose, the training algorithm was chosen to be *trainlm*. We also explored the *K-fold cross validation method* with different training algorithms for the two-layer ANN as shown in Figure 5.5. In most cases, *trainlm* (the curve containing circular dots) gave reasonable correlation values for different number of neurons in the hidden layer. On the other hand, the networks with *trainbr* (the curve containing cross markers) gave higher correlation values for different number of neurons in the hidden layer. However, the training algorithm for *trainbr* is quite slower than the others. As a result, *trainlm* was selected in further experiments.

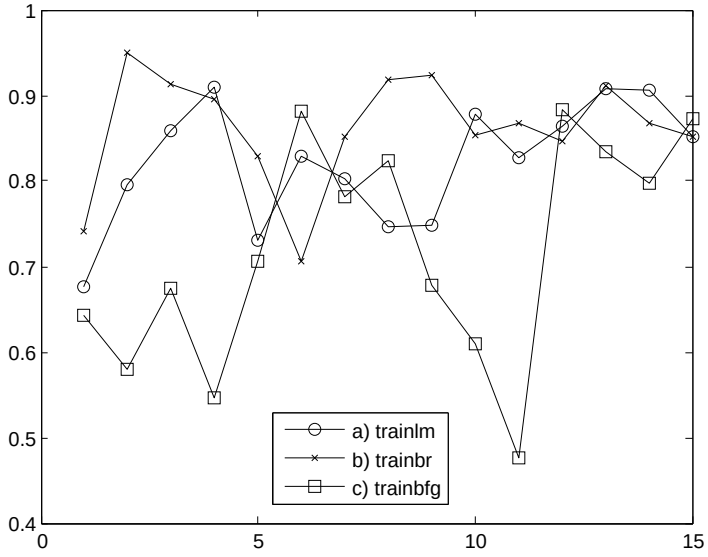


Figure 5.5. *The coefficient of correlation measures for two-layer ANN with different number of neurons. The training methods used in this case are a) trainlm for Levenberg-Marquardt algorithm, b) trainbr for Bayesian Regularization algorithm and c) trainbfg for BFGS Quasi-Newton algorithm.*

After creating the feedforward networks, the experimental dataset were divided into the training set and test set. The training set was used for training the networks with the built-in function `train`. The test set was used for testing the network with `sim` function. For the two-layer network, 4 neurons were selected for the one hidden layer. At this value, the coefficient of correlation computed by *5-fold cross validation*

method was 0.91. For the three-layer network, 12 and 3 were selected for the two hidden layers respectively. In this case, the coefficient of correlation for the network was 0.94.

Figure 5.6 and Figure 5.7 represent the yield prediction curves for the two-layer and the three-layer ANN models, respectively. In Figure 5.6, the effect on yield responses is maximum for higher concentration values of NH_4Cl and lower concentration values of K_2HPO_4, KH_2PO_4, $MgCl_2.6H_2O$ and KCl. The plots also indicate that the interactions between K_2HPO_4 and other components have little or no significance on yield. The interactions among other variables (KH_2PO_4 - $MgCl_2.6H_2O$, KH_2PO_4 - KCl, and $MgCl_2.6H_2O$ - KCl) have negative effect. Likewise, we can see the similar effect of the components on yield responses in Figure 5.7.

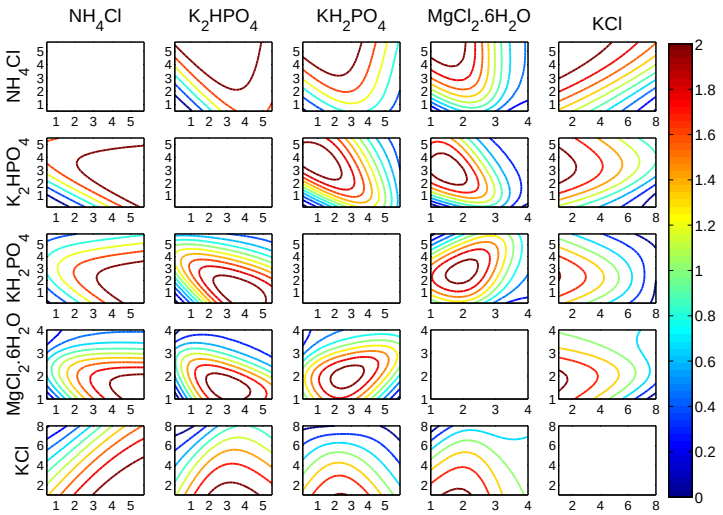


Figure 5.6. *The effects of the yield for pairwise components using the two-layer ANN model with one hidden layer. The yield responses are presented by different colors according to the colorbar. The plots in the diagonal (that is, components are plotted against themselves) are left empty.*

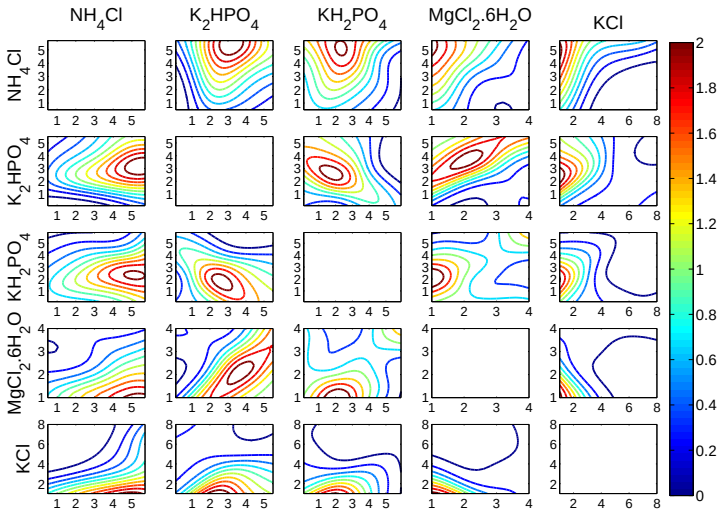


Figure 5.7. *The effects of the yield for pairwise components using the three-layer ANN model with two hidden layers. The yield responses are presented by different colors according to the colorbar. The plots in the diagonal (that is, components are plotted against themselves) are left empty.*

5.2 Performance of the models

For predicted models, the association or correlation between the actual and predicted yield responses can be studied by LOOCV described in Algorithm 2 (see page 18). In other words, we may examine how the models can perform for unseen observations. Table 5.1 summarizes the coefficient of correlation ρ for all prediction models developed in this work. Moreover, whether a model is appropriate for the dataset being analyzed can be studied from a comparison plot of the observed yields against the predicted ones as shown in Figure 5.8 and Figure 5.9. The regression lines, drawn in Figure 5.8 and Figure 5.9, represent the perfect fit that is, the predicted yields are exactly equal to the actual yields. The deviations from the perfect fit are measured for the experimental dataset using the coefficient of correlation ρ. For instance, the value of ρ for two-layer ANN model is 0.92, which is quite close to 1. This means, the yielded two-layer ANN model indicates a good fit for the experimental dataset. The value of ρ is 0.91 for three-layer ANN model.

We also used Fisher's z-transformation [61] to show the differences of statistical significance between the performance of the best model (in this case two-layer ANN) and that of the other models. The results are summarized in Table 5.1. As we can see from the *one-tailed p-value* approach that the difference between two-layer ANN and the other models (both Lasso and MLR) is statistically significant at 95% confidence level. However, there is no difference in performance between two-layer ANN and that of the three-layer ANN for this dataset.

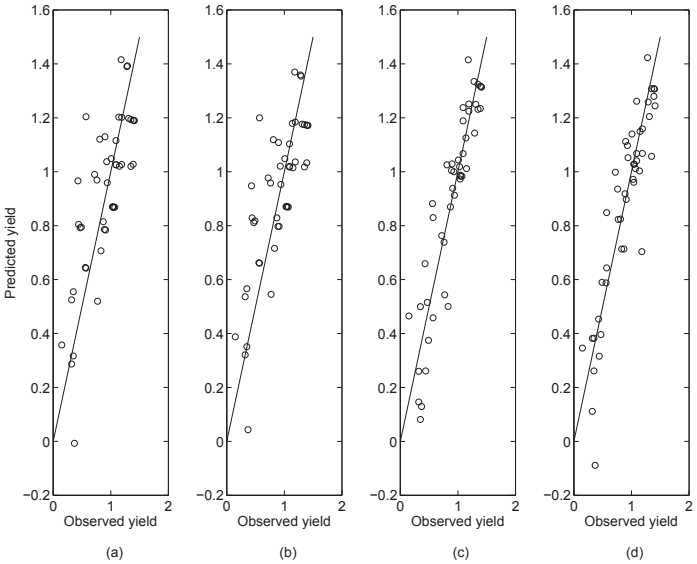


Figure 5.8. *Comparison of prediction performances of (a) linear MLR, (b) linear Lasso, (c) Two-layer ANN, and (d) Three-layer ANN models for experimental dataset.*

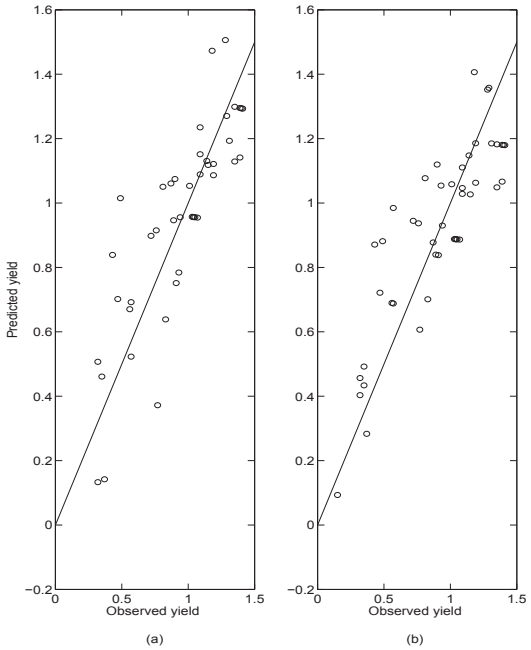


Figure 5.9. *Comparison of prediction performances of (a) quadratic MLR, and (b) quadratic Lasso models for transformed experimental dataset.*

Table 5.1. *The coefficient of correlation ρ and p-value for different prediction models. The ρ was computed using leave-one-out cross validation described in Algorithm 2. The one-tailed p-value test was performed utilizing the tool in [62]. Here, the ρ of two-layer ANN was compared with those of the others. Thus p-value is not available for two-layer ANN.*

Models	ρ	*p-value*
Linear MLR	0.78	0.0049
Quadratic MLR	0.37	0
Linear Lasso	0.78	0.0049
Quadratic Lasso	0.82	0.0202
Two-layer ANN	0.92	-
Three-layer ANN	0.91	0.3859

Although the structures of the linear models are different for both MLR and Lasso (see Equation (5.2) and Equation (5.5)), the values of ρ are similar (0.78). The performances of these models are not as good as those of the ANN models, since both MLR and Lasso are unable to explore the nonlinear behavior in the experimental dataset. Therefore, transformed dataset have also been studied to develop the quadratic models (see Equation (5.4) and Equation (5.6)).

5.3 Optimization of the yields

Once the models are developed, GA described in Chapter 3 can be used to optimize and determine the maximum achievable H_2 yield. Moreover, we can also explore the optimum values of the components where this maximum can be obtained. In this study, the models obtained in Section 5.1 were used as the fitness functions in GA optimization. The GA Matlab Toolbox [63] was used to create the population and search for the optimum individuals. Since GA in Matlab is a minimization toolbox, the fitness function must be negative for maximization. The size of population was chosen to be 20. For parent selection, stochastic universal sampling approach was used. Here, the members in the population are selected according to rank selection and elitism methods. During reproduction, two-point crossover operation was chosen. The algorithm would be terminated either after 500 generations or no improvement in the best fitness value is found. These options are summarized in Table 5.2.

Table 5.2. *The design parameter options of GA.*

Parameter	Value
Population size	20
Elite count	2
Generations	500
Number of independent variables	5
Lower bounds on variables	0.50, 0.10, 0.10, 1.00, 1.00
Upper bounds on variables	5.70, 5.60, 5.90, 4.00, 8.00
Population type	Double vector
Scaling function	fitscalingrank
Selection function	selectionstochunif
Crossover function	crossovertwopoint
Creation function	gacreationlinearfeasible
Mutation function	mutationadaptfeasible

The maximum H_2 yield predicted for all the models developed in this study are listed in Table 5.3. Consider the linear MLR model. After optimization, the maximum H_2 yield for this model was predicted to be 2.24 mol-H_2/mol-glycerol$_{consumed}$. The concentrations of NH_4Cl, K_2HPO_4, KH_2PO_4, $MgCl_2.6H_2O$ and KCl components required for the maximum production were predicted to be 5.70 g/L, 0.57 g/L, 0.12 g/L, 1.00 g/L, and 7.99 g/L, respectively. In case of Lasso and ANN, the values of maximum yields are quite closer though the values were lower than that of the linear MLR model. However, the maximum yield found in the quadratic MLR model was significantly lower than those of the others (0.61 mol-H_2/mol-glycerol$_{consumed}$).

Table 5.3. *The optimum values determined by GA.*

	NH_4Cl	K_2HPO_4	KH_2PO_4	$MgCl_2.6H_2O$	KCl	Yield
Methods	x_1	x_2	x_3	x_4	x_5	y
Linear MLR	5.70	0.57	0.12	1.00	7.99	2.24
Quadratic MLR	0.56	1.03	0.52	1.00	1.04	0.61
Linear Lasso	5.69	1.90	0.11	1.08	3.38	1.51
Quadratic Lasso	5.70	1.67	0.10	1.01	1.18	1.49
Two-layer ANN	5.23	3.29	0.11	1.00	1.47	1.49
Three-layer ANN	5.42	3.15	0.61	1.02	4.19	1.47

In the following section, we will evaluate the prediction capability and the significance of optimal values for each model.

5.4 Analysis of the results

The developed models were used to explore the significance of the five components on the H_2 yield. Consequently, we were able to determine the advantageous control direction of these components. The interaction effects between any two components are illustrated graphically in the contour plots (Figures 5.1, 5.2, 5.3, 5.4, 5.6, and 5.7).

The performance of the predicted models is affected by the nonlinearity of the dataset. Both the linear MLR and the linear Lasso model were not able to fit the nonlinearity of the dataset, resulting in lower performance ($\rho = 0.78$) than that of the others. For two-layer ANN and three-layer ANN, the predicted and the observed responses are correlated strongly (see Figure 5.8). In this case, the values of ρ were 0.92 and 0.91, for two-layer and three-layer ANN models, respectively. Due to the existence of nonlinearity characteristic in the experimental dataset, the regression methods are not sufficient to develop the prediction models. Therefore, the maximum response cannot be perceived inside the examined space, though it would be located in there. Despite the linear structure of Lasso, the prediction performance was increased with the transformed dataset ($\rho = 0.82$). However, the use of transformed dataset in the quadratic MLR was inadequate, which gave a quite poor performance ($\rho = 0.37$).

The subset of experimental dataset in Chapter 4 was used in a separate study [64]. The authors studied MLR, Lasso and random forest methods for H_2 yield maximization. For linear and quadratic Lasso, they showed that the prediction performances were $\rho = 0.60$ and $\rho = 0.69$, respectively. In this work, we were able to achieve improvement of approximately 30% for the linear Lasso and 18.84% for the quadratic Lasso than those of reported by the previous study.

Once the satisfactory models were created, GA was applied for optimization. For the H_2 production examined in this work, the optimum values of the H_2 yield and medium concentrations were obtained by GA. The performance of the GA was affected by the number of design parameters. The results are summarized in Table 5.3. The maximum achievable H_2 yield for this production dataset was 2.24 mol-H_2/mol-glycerol$_{consumed}$, according to the linear MLR model. This maximal H_2 yield was predicted at concentrations of 5.70 g/L, 0.57 g/L, 0.12 g/L, 1.00 g/L and 7.99 g/L for NH_4Cl, K_2HPO_4, KH_2PO_4, $MgCl_2.6H_2O$, and KCl, respectively. The maximal achievable H_2 yields obtained by GA were 0.61 mol-H_2/mol-glycerol$_{consumed}$, 1.51 mol-H_2/mol-glycerol$_{consumed}$, 1.49 mol-H_2/mol-glycerol$_{consumed}$, 1.49 mol-H_2/mol-glycerol$_{consumed}$ and 1.47 mol-H_2/mol-glycerol$_{consumed}$ for quadratic MLR, linear Lasso, quadratic Lasso, two-layer ANN and three-layer ANN, respectively. For both Lasso and ANN model, the small differences in maximal predicted

yield indicate that solutions obtained by both methods are guaranteed to be optimum.

In order to inspect the maximum achievable optima, the yield sensitivity against each medium component was examined. That is, the yield was plotted as a function of the concentration of each medium component. The remaining medium components were set at their corresponding optimum levels, which are listed in Table 5.3. Figure 5.10 and Figure 5.11 represent the yield predicted for the linear and quadratic MLR models, respectively. The curves indicate that optimum yield can be approximately achieved at the same point obtained by GA. However, GA was not able to find the optima for K_2HPO_4 (see Figure 5.10). Similarly, for the linear and quadratic Lasso models (see Figure 5.12 and Figure 5.13), the plots illustrated that optimum levels are achieved at the same point achieved by GA optimization. Since K_2HPO_4 was ignored by both linear and quadratic Lasso models, the yield plot curves for K_2HPO_4 are irrelevant. From Figure 5.14 and Figure 5.15, we can easily observe that the optimum points predicted by the both ANN models matched with those achieved by GA.

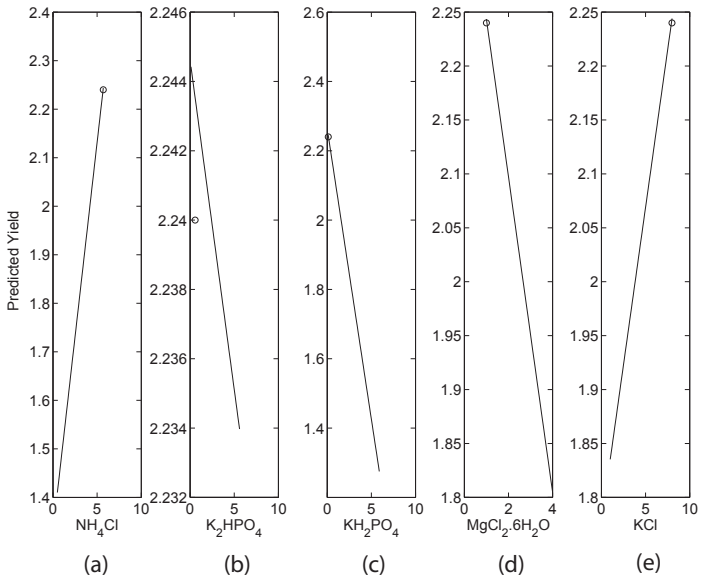


Figure 5.10. *The predicted model is obtained using MLR method for experimental data. The prediction of yields for components (a) NH_4Cl, (b) K_2HPO_4, (c) KH_2PO_4, (d) $MgCl_2.6H_2O$, and (e) KCl. The true optimum yield (2.24 mol-H_2/mol-glycerol$_{consumed}$) obtained using GA is shown in circle. For K_2HPO_4, GA was not able to find the optima. The ranges in X-axis are limited between minimum and maximum concentrations of the corresponding components.*

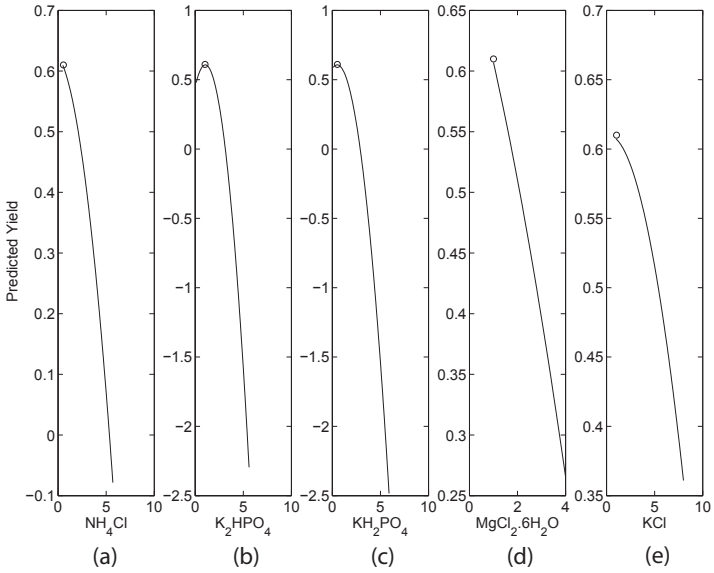


Figure 5.11. *The predicted model is obtained using MLR method for transformed data. The prediction of yields for components (a) NH_4Cl, (b) K_2HPO_4, (c) KH_2PO_4, (d) $MgCl_2.6H_2O$, and (e) KCl. The true optimum yield (0.61 mol-H_2/mol-glycerol$_{consumed}$) obtained using GA is shown in circle. The ranges in X-axis are limited between minimum and maximum concentrations of the corresponding components.*

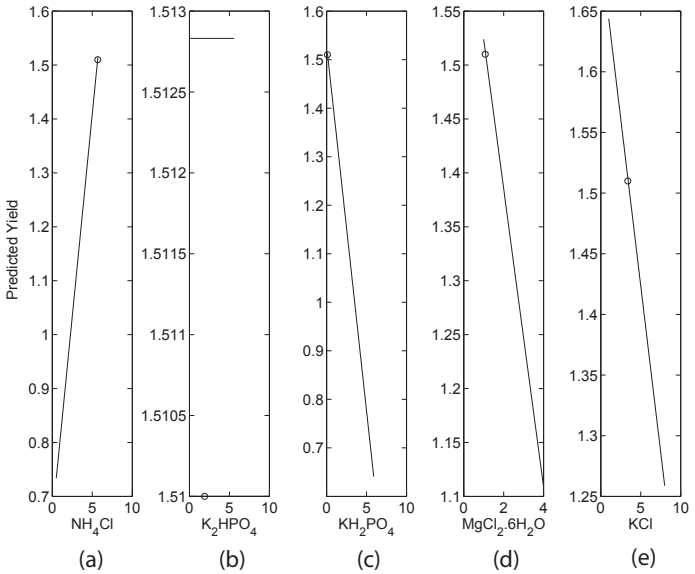


Figure 5.12. *The predicted model is obtained using Lasso method for experimental data. The prediction of yields for components (a) NH_4Cl, (b) K_2HPO_4, (c) KH_2PO_4, (d) $MgCl_2.6H_2O$, and (e) KCl. The true optimum yield (1.51 mol-H_2/mol-glycerol$_{consumed}$) obtained using GA is shown in circle. For K_2HPO_4 and KCl, GA was not able to find the optima. The ranges in X-axis are limited between minimum and maximum concentrations of the corresponding components.*

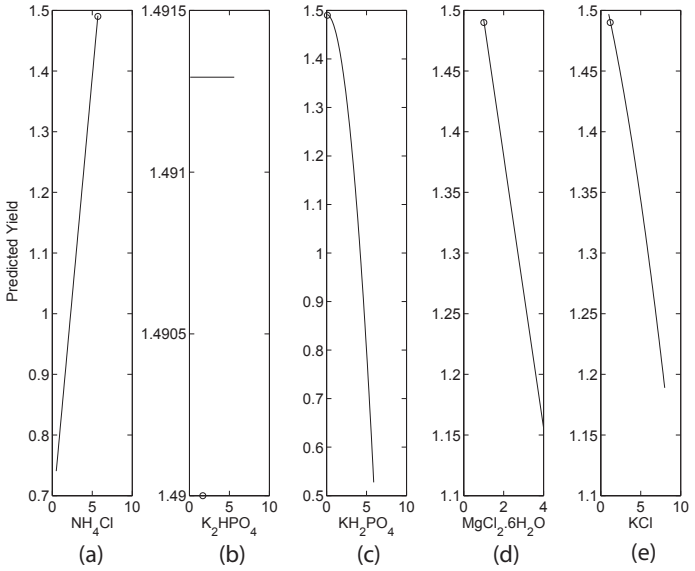


Figure 5.13. *The predicted model is obtained using Lasso method for transformed data. The prediction of yields for components (a) NH_4Cl, (b) K_2HPO_4, (c) KH_2PO_4, (d) $MgCl_2.6H_2O$, and (e) KCl. The true optimum yield (1.49 mol-H_2/mol-glycerol$_{consumed}$) obtained using GA is shown in circle. For K_2HPO_4, GA was not able to find the optima. The ranges in X-axis are limited between minimum and maximum concentrations of the corresponding components.*

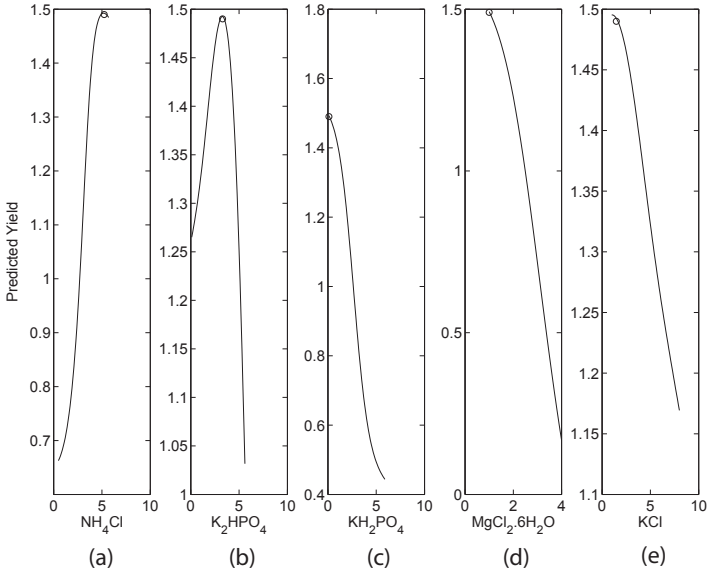


Figure 5.14. *The predicted model is obtained using two-layer ANN method for experimental data. The prediction of yields for components (a) NH_4Cl, (b) K_2HPO_4, (c) KH_2PO_4, (d) $MgCl_2.6H_2O$, and (e) KCl. The true optimum yield (1.49 mol-H_2/mol-glycerol$_{consumed}$) obtained using GA is shown in circle. The ranges in X-axis are limited between minimum and maximum concentrations of the corresponding components.*

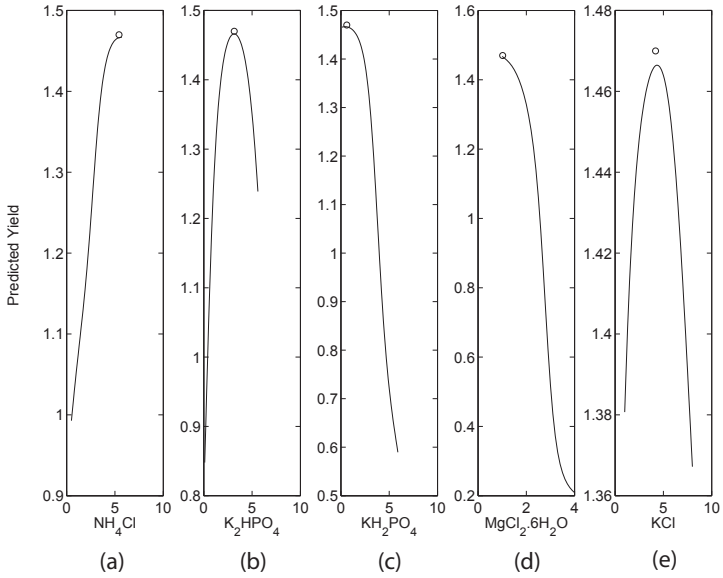


Figure 5.15. *The predicted model is obtained using three-layer ANN method for experimental data. The prediction of yields for components (a) NH_4Cl, (b) K_2HPO_4, (c) KH_2PO_4, (d) $MgCl_2.6H_2O$, and (e) KCl. The true optimum yield (1.47 mol-H_2/mol-glycerol$_{consumed}$) obtained using GA is shown in circle. For KCl, GA was not able to find the optima. The ranges in X-axis are limited between minimum and maximum concentrations of the corresponding components.*

The aforementioned optimized yields except the MLR models are very similar with the results in the statistical design experiments in [55]. The authors in [55] reported that the predicted and empirical experiments for H_2 yield are confirmed to be 1.41 mol-H_2/mol-glycerol$_{consumed}$ and 1.42±0.15 mol-H_2/mol-glycerol$_{consumed}$, respectively. The experiment was conducted at the optimum concentrations of 4.4 g/L, 2.27 g/L, 1.6 g/L, 1.0 g/L and 1.0 g/L for NH_4Cl, K_2HPO_4, KH_2PO_4, $MgCl_2.6H_2O$, and KCl, respectively. The results apparently suggest that the methods used in our present study are effective in modeling and optimizing this kind of bioprocess data.

6. DISCUSSION AND CONCLUSION

This thesis provides a study about bioprocess modeling of H_2 production dataset. The applicability of *Lasso* regression and *artificial neural network* (ANN) in bioprocess data analysis was examined and their performances were benchmarked against *multiple linear regression* (MLR). Moreover, the models were optimized by a *genetic algorithm* (GA) to maximize the H_2 production. The aim was to produce feasible models, study their prediction accuracy, and determine the optimal conditions for maximizing the H_2 production.

MLR is the most common statistical modeling technique, and it is also suitable for prediction problems. In the present study, the MLR model provides a good approximation of the H_2 yield. The *coefficient of correlation* (ρ) of the linear MLR model was 0.78. A quadratic MLR model was also developed by transforming the H_2 production dataset. That yielded prediction accuracy of 0.37. This result demonstrates that the quadratic MLR model was not able to approximate the nonlinear nature of the dataset.

By using Lasso, on the other hand, we achieved the prediction accuracy of 0.78 and 0.82 for linear and quadratic models, respectively. The linear model in MLR obtained the same prediction accuracy as that of the Lasso model ($\rho = 0.78$). However, the sparsity property exhibited in Lasso was able to identify and discard an unnecessary component. According to the model in Equation (5.5), K_2HPO_4 has no effect on H_2 yield, since the coefficient of K_2HPO_4 was zero. Moreover, the prediction accuracy in the quadratic Lasso model was higher ($\rho = 0.82$) than that of the quadratic MLR model ($\rho = 0.37$). Thus, Lasso is as an efficient analysis tool for complicated datasets that include nonlinearly transformed variables.

ANN is a good alternative modeling method. It has the ability to determine any linear or nonlinear relationships between input and output variables. In this thesis, we developed two-layer and three-layer ANN models with prediction accuracy being 0.92 and 0.91, respectively. The observations suggest that ANN models are more accurate in predicting the H_2 production than the two regression methods. However, ANN is quite sensitive in modeling which may lead to overfitting. In this study, the overfitting was avoided by using the *K-fold cross validation* approach.

In order to predict the maximal achievable H_2 yield, the developed models were used as objective functions in the GA. The optimal levels of the five culture medium

components are listed in Table 5.3. The maximal H_2 yield for linear and quadratic MLR models were found to be 2.24 mol-H_2/mol-glycerol$_{consumed}$ and 0.61 mol-H_2/mol-glycerol$_{consumed}$, respectively. For linear and quadratic Lasso models, the maximal H_2 yield were found to be 1.51 mol-H_2/mol-glycerol$_{consumed}$ and 1.49 mol-H_2/mol-glycerol$_{consumed}$, respectively. For two-layer and three-layer ANN, the maximal H_2 yield were found to be 1.49 mol-H_2/mol-glycerol$_{consumed}$ and 1.47 mol-H_2/mol-glycerol$_{consumed}$, respectively. These results suggest that the maximal predicted yields for Lasso and ANN models were lower than that of the linear MLR model. However, the experimentally identified maximal H_2 yield was 1.42±0.15 mol-H_2/mol-glycerol$_{consumed}$ [55]. In other words, the maximum predicted yield with the linear MLR model was much higher than the highest experimental observation in [55]. This questions the feasibility of the linear MLR model in predicting the maximal H_2 yield.

Both Lasso and ANN models have adequate prediction capability of medium optimization in bioprocess development. It is unlikely to choose one method on all circumstances, since the choice depends on the nature of the dataset available and the goals of a model developer. ANN may be particularly useful when the complex nonlinear relationships exist in the dataset. On the other hand, Lasso remains a clear choice when the primary goal is to interpret the model easily or to find the causal relationships between input and output. Further research will be acquired to justify the correlation between predicted and experimental optimal H_2 yield. It would also be of interest to study these methods with different types of bioprocesses.

REFERENCES

[1] D. B. Levin, L. Pitt, and M. Love, "Biohydrogen production: prospects and limitations to practical application," *International Journal of Hydrogen Energy*, vol. 29, no. 2, pp. 173–185, 2004.

[2] R. Mangayil, M. Karp, and V. Santala, "Bioconversion of crude glycerol from biodiesel production to hydrogen," *International Journal of Hydrogen Energy*, vol. 37, no. 17, pp. 12198 – 12204, 2012.

[3] J. R. Almeida, L. C. Fávaro, and B. F. Quirino, "Biodiesel biorefinery: opportunities and challenges for microbial production of fuels and chemicals from glycerol waste," *Biotechnology for Biofuels*, vol. 5, no. 1, pp. 1–16, 2012.

[4] S. J. Sarma, S. K. Brar, E. B. Sydney, Y. L. Bihan, G. Buelna, and C. R. Soccol, "Microbial hydrogen production by bioconversion of crude glycerol: A review," *International Journal of Hydrogen Energy*, vol. 37, no. 8, pp. 6473 – 6490, 2012.

[5] G. P. da Silva, M. Mack, and J. Contiero, "Glycerol: A promising and abundant carbon source for industrial microbiology," *Biotechnology Advances*, vol. 27, no. 1, pp. 30–39, 2009.

[6] I. Orhan, *Biotechnological Production of Plant Secondary Metabolites*, Bentham Science Publishers, 2012.

[7] D. C. Montgomery, *Design and analysis of experiments*, vol. 7, Wiley New York, 1984.

[8] C. Pan, Y. Fan, Y. Xing, H. Hou, and M. Zhang, "Statistical optimization of process parameters on biohydrogen production from glucose by *Clostridium* sp. fanp2," *Bioresource technology*, vol. 99, no. 8, pp. 3146–3154, 2008.

[9] Y. Nagata and K. H. Chu, "Optimization of a fermentation medium using neural networks and genetic algorithms," *Biotechnology letters*, vol. 25, no. 21, pp. 1837–1842, 2003.

[10] C. F. Mandenius and A. Brundin, "Bioprocess optimization using design-of-experiments methodology," *Biotechnology progress*, vol. 24, no. 6, pp. 1191–1203, 2008.

[11] S. S. Sablani, A. K. Datta, M. S. Rahman, and A. S. Mujumdar, *Handbook of food and bioprocess modeling techniques*, CRC Press, 2006.

[12] N. H. Bingham, N. Bingham, and J. M. Fry, *Regression: Linear models in statistics*, Springer, 2010.

[13] S. M. Stigler, "Mathematical statistics in the early states," *The Annals of Statistics*, vol. 6, no. 2, pp. 239–265, 1978.

[14] T. Hastie, R. Tibshirani, and J. Friedman, *The elements of statistical learning: data mining, inference, and prediction*, vol. 2nd edition, Springer, New York, 2009.

[15] A. E. Hoerl and R. W. Kennard, "Ridge regression: biased estimation for nonorthogonal problems," *Technometrics*, vol. 42, no. 1, pp. 80–86, 2000.

[16] R. Tibshirani, "Regression shrinkage and selection via the lasso," *Journal of the Royal Statistical Society. Series B (Methodological)*, vol. 58, no. 1, pp. 267–288, 1996.

[17] A. N. Tikhonov, "Solution of incorrectly formulated problems and the regularization method," in *Doklady Akademii Nauk SSSR*, 1963, vol. 151, pp. 501–504.

[18] A. N. Tikhonov and V. Y. Arsenin, *Solutions of ill-posed problems*, 1977.

[19] M. Schmidt, G. Fung, and R. Rosales, "Fast optimization methods for l_1 regularization: A comparative study and two new approaches," in *Machine Learning: ECML 2007*, vol. 4701, pp. 286–297. Springer, 2007.

[20] B. Efron, T. Hastie, I. Johnstone, and R. Tibshirani, "Least angle regression," *The Annals of statistics*, vol. 32, no. 2, pp. 407–499, 2004.

[21] H. Xu, C. Caramanis, and S. Mannor, "Robust regression and lasso," *IEEE Transactions on Information Theory*, vol. 56, no. 7, pp. 3561–3574, 2010.

[22] F. Rosenblatt, "The perceptron: a probabilistic model for information storage and organization in the brain," *Psychological review*, vol. 65, no. 6, pp. 386–408, 1958.

[23] F. Rosenblatt, *Principles of neurodynamics*, Spartan Books, New York, 1962.

[24] S. N. Diggavi, J. J. Shynk, I. Engel, and N. J. Bershad, "On the convergence behavior of rosenblatt's perceptron learning algorithm," in *1992 Conference Record of The Twenty-Sixth Asilomar Conference on Signals, Systems and Computers*. IEEE, 1992, vol. 2, pp. 852–856.

[25] S. N. Diggavi, J. J. Shynk, and N. J. Bershad, "Convergence models for rosenblatt's perceptron learning algorithm," *IEEE Transactions on Signal Processing*, vol. 43, no. 7, pp. 1696–1702, 1995.

[26] H. Demuth and H. M. Beale, *Neural Network Toolbox For Use with Matlab*, Mathworks, Natick, Mass., 1998.

[27] S. Weisberg, *Applied linear regression*, vol. 3rd edition, Wiley. com, 2005.

[28] R. H. Myers, *Classical and modern regression with applications*, vol. 2nd edition, Duxbury Press Belmont, CA, 1990.

[29] D. J. Hand, "Discrimination and classification," *Wiley Series in Probability and Mathematical Statistics*, vol. 1, 1981.

[30] G. McLachlan, *Discriminant analysis and statistical pattern recognition*, Wiley. com, 1992.

[31] S. Weiss and C. Kulikowski, *Computer systems that learn*, San Mateo, CA: Morgan Kaufmann, 1991.

[32] B. Widrow and M. E. Hoff, "Adaptive switching circuits," *IRE Wescon Convention Record*, vol. 4, pp. 96–104, 1960.

[33] D. W. Hosmer Jr, S. Lemeshow, and R. X. Sturdivant, *Applied logistic regression*, vol. 3rd edition, Wiley. com, 2013.

[34] J. C. Principe, N. R. Euliano, and W. C. Lefebvre, *Neural and adaptive systems: fundamentals through simulations*, vol. 1st edition, John Wiley & Sons, Inc., 1999.

[35] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural networks*, vol. 2, no. 5, pp. 359–366, 1989.

[36] H. White, *Artificial neural networks: approximation and learning theory*, Blackwell Publishers, Inc., Cambridge, MA, USA, 1992.

[37] L. Devroye and T. Wagner, "Distribution-free inequalities for the deleted and holdout error estimates," *IEEE Transactions on Information Theory*, vol. 25, no. 2, pp. 202–207, 1979.

[38] O. Nelles, *Nonlinear system identification: from classical approaches to neural networks and fuzzy models*, Springer, 2001.

[39] R. Horst, P. M. Pardalos, and H. E. Romeijn, *Handbook of global optimization*, vol. 2, Kluwer Academic Publishers, 2002.

[40] J. H. Holland, *Adaptation in Natural and Artificial Systems*, University of Michigan Press, Ann Arbor, MI, 1975.

[41] M. Melanie, "An introduction to genetic algorithms," *Cambridge, Massachusetts London, England, Fifth printing*, vol. 3, 1999.

[42] R. A. Caruana, "Representation and hidden bias: Gray vs. binary coding for genetic algorithms," in *Proceedings of the Fifth International Conference on Machine Learing Ann Arbor, Mich.*, 1988, pp. 153–161.

[43] D. E. Goldberg, *Genetic algorithms in search, optimization, and machine learning*, Addison-Wesley, 1989.

[44] C. Z. Janikow and Z. Michalewicz, "An experimental comparison of binary and floating point representations in genetic algorithms.," in *ICGA*, 1991, pp. 31–36.

[45] J. E. Baker, "Reducing bias and inefficiency in the selection algorithms," in *Proceedings of the 2nd International Conference on Genetic Algorithms*, 1987.

[46] K. A. De Jong, *Analysis of the behavior of a class of genetic adaptive systems*, Ph.D. thesis, Dept. Comput. Sci., Univ. Michigan, 1975.

[47] R. L. Haupt and S. E. Haupt, *Practical genetic algorithms*, John Wiley & Sons, 2004.

[48] J. E. Baker, "Adaptive selection methods for genetic algorithms," in *Proceedings of the 1st International Conference on Genetic Algorithms*. L. Erlbaum Associates Inc., 1985, pp. 101–111.

[49] S. Forrest, "Scaling fitnesses in the genetic algorithm," *Documentation for Prisoners Dilemma and Norms Programs That Use the Genetic Algorithm. Unpublished manuscript*, 1985.

[50] D. E. Goldberg, "A note on boltzmann tournament selection for genetic algorithms and population-oriented simulated annealing," *Complex Systems*, vol. 4, no. 4, pp. 445–460, 1990.

[51] J. Y. Suh and D. Van Gucht, *Distributed genetic algorithms*, 1987.

[52] T. Back, D. B. Fogel, and Z. Michalewicz, *Handbook of evolutionary computation*, IOP Publishing Ltd., 1997.

[53] G. Sywerda, "Uniform crossover in genetic algorithms," in *Proceedings of the third international conference on Genetic algorithms*. Morgan Kaufmann Publishers Inc., 1989, pp. 2–9.

[54] W. M. Spears and K. D. De Jong, "On the virtues of parameterized uniform crossover," Tech. Rep., DTIC Document, 1995.

[55] R. Mangayil, T. Aho, M. Karp, and V. Santala, "Improved bioconversion of crude glycerol to hydrogen by optimizing selected media components: A statistical approach," Unpublished, 2013.

[56] R. L. Plackett and J. P. Burman, "The design of optimum multifactorial experiments," *Biometrika*, vol. 33, no. 4, pp. 305–325, 1946.

[57] G. E. Box and D. Behnken, "Some new three level designs for the study of quantitative variables," *Technometrics*, vol. 2, no. 4, pp. 455–475, 1960.

[58] N. R. Draper, ""ridge analysis" of response surfaces," *Technometrics*, vol. 5, no. 4, pp. 469–479, 1963.

[59] G. E. Box and K. Wilson, "On the experimental attainment of optimum conditions," *Journal of the Royal Statistical Society. Series B (Methodological)*, vol. 13, no. 1, pp. 1–45, 1951.

[60] J. Friedman, T. Hastie, and R. Tibshirani, "Lasso (l_1) and elastic-net regularized generalized linear models," http://www-stat.stanford.edu/∼tibs/glmnet-matlab/, 2010.

[61] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, *Numerical Recipes in C: The Art of Scientific Computing*, Cambridge Univ. Press, 1997.

[62] R. Lowry, "Significance of the difference between two correlation coefficients," `http://www.vassarstats.net/rdiff.html`, 2001.

[63] A. Chipperfield, P. Fleming, H. Pohlheim, and C. Fonseca, *Genetic algorithm toolbox for use with MATLAB*, Citeseer, 1994.

[64] S. S. Hassan, M. Farhan, R. Mangayil, H. Huttunen, and T. Aho, "Bioprocess data mining using regularized regression and random forests," *BMC Systems Biology*, vol. 7, no. Suppl 1, pp. S5, 2013.